Київський національний університет імені Тараса Шевченка

На правах рукопису

Паскевич Андрій Юрійович

УДК 004.832.3

Засоби формалізації математичних знань та міркувань: теоретичні та практичні аспекти

01.05.01 — теоретичні основи інформатики та кібернетики

ДИСЕРТАЦІЯ

на здобуття вченого ступеня
кандидата фізико-математичних наук

Науковий керівник
Донченко Володимир Степанович,
кандидат фізико-математичних наук, професор

Київ — 2005

ЗМІСТ

ВСТУП

i.1. Загальна характеристика роботи

Актуальність теми. З кожним роком невпинно розширюється сфера застосування комп'ютерних програмних систем. Без використання новітніх інформаційних технологій сьогодні неможливо уявити успішне вирішення економічних і військових, банківських і технічних задач. Це висуває жорсткі вимоги щодо безпечності та надійності сучасного програмного забезпечення, призначеного для критично відповідальних задач, ці якості мають бути строго доведені, а не встановлені емпірично. Тому необхідність в ефективних та зручних системах автоматичного доведення виходить за межи суто наукових застосувань: при розробці, наприклад, систем реального часу, зайнятих у промисловості та транспорті, або захищених протоколів зв'язку, такі системи відіграють ключову роль. Необхідно відмітити, що складність цієї проблеми робить неможливою повну автоматизацію її вирішення. Тому надзвичайно важливою виявляється взаємодія людини з машиною, необхідна для розв'язання задач, що вимагають як обчислювальної потужності для громіздких рутинних побудов, так і інтелекту разом з інтуїцією, для вибору потрібного напрямку розв'язання.

Незалежно від їхнього походження, твердження, що мають бути перевірені, є, фактично, математичними твердженнями, сформульованими в деякій формальній теорії. Тому доцільно розглядати задачу перевірки цих тверджень не в сенсі побудови виведення в тій чи іншій формальній дедуктивній системі, а як задачу проведення математичного міркування. Середою для такого міркування виступатиме математичний текст, подібний до "природних" текстів, що публікуються в журналах та підручниках. Такий підхід дає можливість користувачеві співпрацювати з машиною в зручному та звичному для себе оточенні: запроваджувати ескізи доведень, відслідковувати шлях міркування програми, коригувати його. Важливість задачі вивчення та формалізації традиційних засобів подання математичного знання та схем математичного міркування для

успішного широкого застосування інформаційних технологій і визначає актуальність даної дисертаційної роботи.

Зв'язок з науковими програмами, планами, темами. Дисертаційне дослідження виконувалося в рамках наукової теми "Логіко-математичні та програмологічні засоби інформаційних технологій" (державний реєстраційний номер 01БФ015-07), яка виконувалась на факультеті кібернетики Київського національного університету імені Тараса Шевченка. Робота продовжує цикл досліджень за програмою "Алгоритм Очевидності" (АО) [1, 2, 3], започаткованою В.М. Глушковим на початку 1970-х років в Інституті кібернетики АН УРСР.

Мета і завдання дослідження. Головною метою роботи є побудова системи комп'ютерних засобів автоматичної перевірки коректності формалізованого математичного тексту. В роботі розробляється оригінальна дворівнева архітектура процедури автоматичного доведення, де верхній рівень ("міркувальник") виконує високорівневі кроки доведення, імітуючи традиційні прийоми математичного міркування, а нижній рівень "закриває" породжені підцілі за допомогою комбінаторної процедури пошуку виведення в деякій формальній дедуктивній системі.

Відповідно до вказаної мети визначені основні завдання роботи:

- запропонувати формальну мову для подання математичних текстів;
- сформулювати систему понять, в рамках якої можна визначити коректність тексту, записаного цією мовою;
- розробити "інструментарій" для перевірки коректності: комбінаторні процедури пошуку виведення, а також еврістичні методи доведення, що використовуватимуться на верхньому рівні системи.

Запропоновані рішення мають бути теоретично обґрунтовані та апробовані на практиці у серії експериментів.

Методика дослідження полягає у проведенні аналізу природних математичних текстів, вивчення засобів подання та міркування, що використовуються в них, та побудові формальних аналогів цих засобів. Для теоретичного обґрунтування запропонованих рішень використовувався апарат математичної логіки та теорії логічного виводу.

Описана в дисертаційній роботі мова запису математичних текстів ForTheL побудована на принципах, що закладені в мові TL, розробленій К.П. Вершиніним [4, 5].

В основі табличних цілекерованих числень, що вивчаються в роботі, лежить процедура породження вспоміжних цілей (ПВЦ), запропонована Ф.В. Ануфрієвим [6, 7] та розвинута О.В. Лялецьким та А.І. Дегтярьовим [8, 9]. Поняття допустимої підстановки [10, 11] було введено О.В. Лялецьким з метою оптимізації перебору відносно різних порядків застосування кванторних правил в разі відмовлення від сколемізації.

Для доведення повноти запропонованого у третьому розділі роботи цілекерованого табличного числення для класичної логіки першого порядку з рівністю було використано метод елімінації рівності та метод трансформації виведень.

Наукова новизна одержаних результатів. На базі мови TL розвинуто формальну мову подання математичних текстів ForTheL, що є близькою за синтаксисом до природної англійської мови і може бути перекладена в мову першого порядку. Сформульовано поняття коректності тексту в мові ForTheL.

Запропоновано оригінальне поняття локальної істинності твердження в деякій позиції всередині формули першого порядку та досліджено його властивості. Зокрема, доведено, що коли еквівалентність двох формул є локально істинною в заданій позиції, ці дві формули є взаємозамінюваними в цій позиції. Поняття локальної істинності дозволяє обґрунтувати коректність різноманітних перетворень складних формул, таких як розкриття визначень або додання допоміжних тверджень всередині формули. Також в роботі запропонована процедура породження “відомостей”

про окремі входження термів в досліджуваний ForTheL-текст; ці відомості є літерами, локально істинними в відповідному входженні, зокрема, вони несуть інформацію про тип терму. На базі цього апарату в роботі сформульовано декілька високорівневих схем доведення, які становлять "інструментарій" міркувальника.

Набули розвитку дослідження цілекерованих процедур пошуку виведення, що використовуються в програмі "Алгоритм Очевидності". Дано нове доведення коректності поняття допустимої підстановки. Викладено цілекероване табличне числення, що оперує несколемізованими формулами. Доведено, що виведення в цьому численні можуть бути перетворені у виведення в класичному табличному диз'юнктному численні Model Elimination. Також доведена можливість зворотнього перетворення. Запропоновано нове табличне цілекероване числення з лінивою парамодуляцією для класичної логіки першого порядку з рівністю, доведено його повноту.

Практичне значення. Результати дисертаційної роботи дозволили реалізувати систему автоматичного доведення САД (Система Автоматизації Дедукції), призначену для обробки формальних математичних текстів, зокрема, для перевірки їх коректності. Описана у роботі мова ForTheL може бути застосована як мова формалізації в багатьох задачах: верифікації специфікацій програмного та апаратного забезпечення, верифікації протоколів передачі даних, для створення баз математичного знання, для комунікації комп'ютерних математичних сервісів. Введене у роботі поняття локально істинного твердження може бути використано для формалізації природних прийомів математичного міркування. Техніка генерації літерних відомостей про входження термів може застосовуватись для скорочення пошуку доведення. Запропоноване цілекероване табличне числення з лінивою парамодуляцією може бути застосоване як базове числення при побудові універсального пруверу (програми автоматичного доведення) для логіки першого порядку з рівністю.

Особистий внесок здобувача. Здобувачем особисто була розроблена граматика мови ForTheL, в якій формалізовано фрагмент англійскої мови. Синтаксис ForTheL-речень було розширено (порівняно з TL) багатьма новими оборотами (наприклад, негативний універсальний квантор "no" та дієслово "to have" в твердженнях типу `the empty set has no elements`), дозволяється символьна нотація, синонімія. Запропоновано оригінальний синтаксис і семантику доведень за різними схемами.

Здобувачеві належать оригінальне формулювання цілекерованого табличного числення в стилі ПВЦ та нове доведення коректності і повноти такого числення шляхом побудови рекурсивної процедури перетворення виведень в цьому численні в виведення в класичному численні Model Elimination; таким чином, встановлено зв'язок між двома підходами для класичної логіки першого порядку. Цілекероване табличне числення з правилом лінивої парамодуляції є оригінальною розробкою здобувача.

В роботі [12] здобувачем проведене порівняння двох запропонованих цілекерованих числень з точки зору побудови мінімальних виводів (розділ "Порівняння числень"). Для обох числень знайдено класи задач пошуку виводу, для яких мінімальне дерево виводу у одному численні має лінійний обсяг, а у іншому численні — експоненційний, по відношенню до обсягу задачі.

В роботі [13] здобувачем виконано програмну реалізацію системи САД, йому належить наведений у статті опис системи, а також процедури верифікації математичного тексту (розділи "Архитектура САД", "Верификация в САД").

В тезах доповідей [14] та [15] К.П. Вершиніну та О.В. Лялецькому належать постановка задачі та участь в обговоренні результатів.

В тезах доповіді [16] здобувачеві належать опис архітектури системи, мови ForTheL, демонстрація процедури верифікації тексту на прикладі доведення теореми Рамсея (розділи "System for Automated Deduction", "Linguistic tools of SAD", "Text verification in SAD").

Апробація результатів дисертації. Основні результати докладалися на наукових семінарах в Інституті кібернетики НАН України, на факультеті кібернетики Київського національного університету імені Тараса Шевченка, в лабораторії LACL університету Париж 12 (Франція), в дослідницькій групі OMEGA Саарландського університету (Німеччина), в лабораторії LORIA (Франція), в групі теоретичної інформатики та логіки Віденського технічного університету (Австрія), а також на наступних конференціях та семінарах:

- Міжнародний семінар "Rewriting techniques and efficient theorem proving" (Київ, 2000) [17];
- Міжнародна конференція "Information Theories and Applications 2000" (Варна, Болгарія, 2000) [14];
- Міжнародний семінар "STRATEGIES 2001" (Сієна, Італія, 2001) [18];
- Міжнародний семінар "Implementation Technology for Computational Logic Systems" (Піза, Італія, 2003) [15];
- Міжнародна конференція "Mathematical Knowledge Management 2004" (Білосток, Польща, 2004) [16].

Публікації. Основні результати дисертації опубліковано у 7 роботах, з яких 4 — статті у фахових збірниках наукових праць [19, 12, 20, 13], 3 — тези міжнародних конференцій [14, 15, 16].

i.2. Огляд сучасного стану галузі

Огляд існуючих на цей день підходів та систем в галузі автоматичного пошуку доведення здається доцільним провести з точки зору трьох характеристик: стилю формалізації задачи, що пропонується користувачеві, спосіб подання доведення, який вимагає система, та рівень деталізації доведень, що підтримується в ній. В цьому підрозділі ми намагаємось показати місце програми АО та системи САД в цьому "трьохвимірному просторі".

Стиль формалізації. Під цим ми розуміємо вибір базових математичних відомостей, характер визначень (чи є вони переважно рекурсивними), спосіб проведення міркувань (чи мають вони бути конструктивними, або строго типізованими, або заснованими на обчисленні), і так далі. Очевидно, що стиль формалізації визначається насамперед базовою логікою системи і фундаментальними теоріями, що використовуються в формалізаціях.

Сьогодні, два головних тренда ґрунтуються на логиці вищого порядку (теорії типів) та логиці першого порядку (теорії множини). Типи використовуються в більшості сучасних систем автоматизації доведень, зокрема, в системах Isabelle/HOL [21], Coq [22], ΩMEGA [23], PVS [24], HOL [25], Automath [26], λCLAM [27], та інших. Теоретико-типовий підхід схиляється до рекурсивних визначень, індуктивно заданих доменів, і, зокрема, є цілком природним для формалізації задач програмування або хардверної інженерії. Для традиційної математики цей стиль виявляється не дуже зручним [28] — не заперечуючи того, що в більшості згаданих систем накопичено багаті бібліотеки суто математичного знання.

З іншого боку опиняється відома система Mizar [29], що заснована на класичній логіці першого порядку та теорії множин Тарського-Гротендіка. Цій підхід відповідає традиційному стилю викладання математики, а бібліотека системи Mizar є, на цей час, найбільш розвинутою колекцією формалізованих математичних текстів.

Класична логіка першого порядку лежить і в основі системи САД. Ми не використовуємо ту чи іншу аксиоматичну теорію множин (або іншу фундаментальну теорію) як загальний ґрунт для формалізації. Натомість, автор формального тексту самостійно визначає сукупність вихідних фактів для проблеми, що розглядається, і обирає потрібний йому рівень "розвинутості" базових концепцій. Інакше кажучи, система САД не вирощує одне високе дерево, а працює з багатьма зеленими кущами.

Спеціальний вид логіки було прийнято в прувері Nqthm [30], та в його "спадкоємці", системі ACL2 [31]. Ці системи базуються на безквантворній

логіці першого порядку та теорії індуктивно заданих структур даних. Така основа досить схожа на теоретико-типовий підхід: вона є зручною для формалізації міркувань по індукції над скінчено породженими областями (натуральні числа, списки, дерева) з використанням рекурсивних функцій.

Проект Theorema [32] працює в класичній логіці предикатів вищого порядку, збагаченій спеціальними "кортежними" змінними та зв'язуючими конструкціями: лямбда-абстракція, сума, границя та іншими.

Спосіб доведення. Інша важлива характеристика системи автоматичного доведення, це спосіб керування пошуком доведення — по суті, спосіб подання міркувань. Інтерактивні системи найчастіше є тактико-керованими: користувач доводить твердження послідовними інструкціями системі. Ці інструкції, *тактики*, можуть бути примітивними, наприклад, застосувати *modus ponens* чи інше правило виведення, або, навпаки, досить складними, наприклад, застосувати алгоритм спрощення або вирішуючу процедуру. Тактико-керованими системами є Isabelle, PVS, Coq, HOL та інші. Зручність користування такою системою залежить від того, чи запроваджує вона досить стислий набір потужних тактик, яких достатньо для досягнення цілі в більшості випадків.

Системи іншого типу приймають доведення, записані в тій самій мові, що й аксиоми та твердження задачи. Звісно, ця мова повинна включати засоби структуризації для того, щоб організовувати логічні формули в доведення. Отримавши доведення (певного рівню деталізації), система автономно перевіряє кожен крок міркування. CAД і Mizar є системами цього типу; Isabelle, з появою компоненти Isar [33] (мова подання доведень, що імітує структуру природних доведень), також може вважатися тексто-керованою системою.

Хоча в мові Nqthm та ACL2 немає "розділів доведення", ці системи також слід віднести до другого типу: доведення головного твердження в них підтримується і керується сукупністю допоміжних лем, що доводяться перед ним.

Системи перевірки доведень, такі як Automath, є тексто-керованими системами за визначенням, хоча, в Automath доведення, що перевіряється, є одним великим типізованим лямбда-термом, чий тип відповідає цільовому твердженню за ізоморфізмом Каррі-Говарда.

Безумовно, розрізнення тактико-керованих і тексто-керованих систем є досить умовним. Якщо в тактико-керованій системі доводити теореми, застосовуючи лише кроки введення проміжної цілі та кроки автоматичного закриття підцілей, така система постає тексто-керованою: проміжні твердження і є доведенням. З іншого боку, якщо кроки в доведенні, яке ми передаємо тексто-керованій системі, повинні супроводжуватись детальними підказками верифікатору, така система є, по суті, тактико-керованою (зокрема, це випадок Isar).

Окремо стоять системи, що використовують планування доведення, такі як λCLAM, ΩMEGA, та IsaPlanner [34]. В планувальнику доведення користувач не супроводжує задачу власним доведенням, записаним в тексті або закодованим в послідовності застосувань тактик. Натомість, система намагається знайти доведення самостійно, розробляючи план доведення на базі колекції загальних методів доведення, керуючись передумовами застосування методів та різноманітними стратегіями. Звісно, ці методи доведення, правила та стратегії мають бути застосовні для достатньо широких класів проблем.

Рівень деталізації. Дедуктивна потужність системи автоматичного доведення може бути високою чи низькою, в залежності від чого користувач має виконати меншу чи більшу частину доведення самостійно: виміряючи в довжині запроваджених доведень для тексто-керованих систем або в кількості інструкцій для тактико-керованих. Умовно кажучи, серед систем автоматичного доведення можна виділити системи, які будують доведення, і системи, які лише перевіряють запроваджене доведення. Останні приймають лише такі доведення, де кожний крок полягає у застосуванні деякого правила виведення, тобто вичерпно деталізовані. Системою такого типу є Mizar, хоча набір правил виведення в Mizar

досить великий. Automath також є суто “контролером” доведень.

Системи першого типу застосовують автоматичний пошук виведення або методи планування доведень і намагаються закрити лакуни самостійно. Тексто-керованими системами цього типу є САД, Nqthm (ACL2) та Theorema. λClam та Ωmega, планувальники доведень, також відносяться до “шукачів” доведень.

В інтерактивних тактико-керованих системах, арсенал тактик, як правило, може бути розширений, отже “дедуктивна потужність” тут не притаманна системі як такій, і майже будь-яка тактико-керована система може роглядатися як система пошуку доведення. Втім, проведені експерименти з Coq та Isabelle показують, що користувач має суворо додержуватись стилю формалізації, прийнятого в системі. Наприклад, при спробі довести складну математичну теорему, не формулюючи для неї часткових лем, без використання спеціальних тактик та існуючих бібліотек, діалог конструювання доведення швидко стає широко розгалуженим і досить заплутаним.

i.3. Архітектура системи САД

Система Автоматизації Дедукції [15, 35, 36], запроваджує три основних сервіса:

- пошук виведення в класичній логіці першого порядку з рівністю;
- доведення теорем в контексті замкненого математичного тексту;
- верифікація замкненого математичного тексту.

Ми можемо говорити про три режима функціонування САД: режим пошуку виведення, режим доведення теорем та режим верифікації текстів. Кожний режим спирається на попередній: доведення теорем в САД використовує при необхідності автоматичний пошук виведення, а верифікація тексту складається з доведення певної кількості окремих тверджень щодо текста, який перевіряється системою.

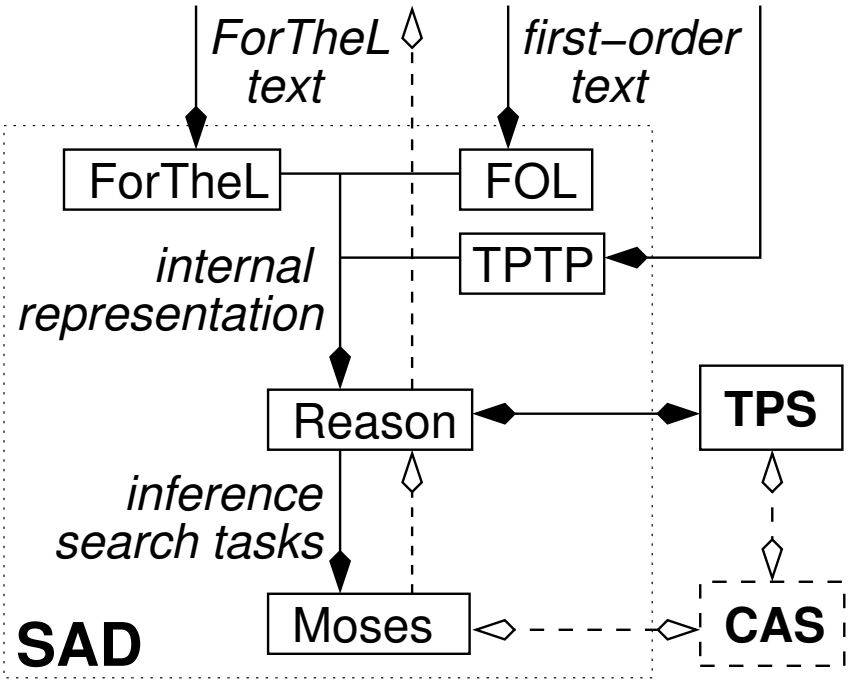


Рис. 1: Архітектура САД

Для того, щоб пояснити архітектуру системи САД, розглянемо її застосування в режимі верифікації. Етапи цього процесу ми будемо пов'язувати з внутрішніми модулями САД на схемі (малюнок 1).

Користувач, якого ми будемо називати *автором*, готує ForTheL-текст, що містить формулювання власне задачі (наприклад, окремої теореми чи декількох теорем), релевантну інформацію (необхідні леми, визначення, аксіоми), та загальні базові відомості, що використовуватимуться впродовж всього тексту (наприклад, основні властивості множин або чисел). Твердження, сформульовані в тексті, можуть супроводжуватись доведенням. Створення ForTheL-тексту є цілком ручною роботою. Перший розділ дисертації присвячено синтаксису та семантиці мови ForTheL. В доданку наведено приклад нетривіальної теореми, записаної і доведеної в ForTheL-тексті.

Підготовлений текст передається на вхід системі САД. Модуль синтаксичного аналізу та трансляції [ForTheL] переводить текст во внутрішнє подання. В цьому поданні речення ForTheL перетворюються в анотовані логічні формули, а сама структура тексту, зокрема, розділів доведень, спрощується і уточнюється: наприклад, в доведенні розбором випадків

твердження вичерпності розбору (диз'юнкція гіпотез випадків) явно додається в доведення. Хоча безпосереднім результатом трансляції є сукупність внутрішніх структур даних системи, ми розглядатимемо цей результат також як ForTheL-текст (нормалізований).

Для перевірки поданого ForTheL-тексту система виділяє з його нормалізованої версії цільові твердження, що мають бути виведені з їхніх логічних попередників в тексті. Таким чином, верифікація тексту зводиться до доведення теорем.

Цільові твердження обробляються по черзі модулем [Reason], *міркувальником* системи САД. Отримавши завдання доведення — цільове твердження разом з сукупністю його попередників в тексті — міркувальник намагається спростити його різними способами: розділити на декілька менших завдань (наприклад, розщіпити кон'юнкцію в цілі або провести розбор випадків), скоротити множину посилок, розкрити визначення, і так далі.

Міркувальник можна розглядати як автоматичний евристичний прувер, який оперує певною сукупністю правил перетворення завдань доведення. Ця сукупність не повинна утворювати повне логічне числення. Задача міркувальника полягає не в тому, щоб самостійно знайти доведення, а в тому, щоб підготувати завдання виведення для *прувера* САД. Останній є автоматичною програмою комбінаторного пошуку виведення в класичній логиці першого порядку, його задача — завершувати доведення розпочаті міркувальником. Якщо прувер не в змозі знайти виведення, міркувальник може продовжити перетворення поточного завдання, спробувати альтернативний шлях або відкинути текст.

Чим потужнішим є арсенал міркувальника, тим більш складні завдання доведення він здатен виконати, і тим стисліші математичні тексти можуть бути перевірені системою. Теоретичні основи побудови міркувальника викладені в другому розділі дисертації.

Власний прувер системи САД, названий [Moses], ґрунтується на спеціальному цілекерованому табличному численні, яке досліджується в третьому розділі даної роботи. Прувер виконує пошук в глибину з обме-

женням глибини, поступовим заглибленням та бектрекінгом. Техніка допустимої підстановки, що використовується в численні прувера, дозволяє відмовитись від сколемізації, а отже зберігти початкову сигнатуру завдання. Таким чином, породжені впродовж пошуку виведення завдання уніфікації (системи рівностей та нерівностей) можуть бути направлені спеціальній розв'язуючій процедурі, наприклад, зовнішній системі комп'ютерної алгебри (computer algebra system, **CAS**).

Ми називаємо [Moses] власним прувером САД, оскільки, за дизайном системи, прувер є відокремленою програмою пошуку виведення, і САД дозволяє використовувати в цій якості зовнішні системи автоматичного доведення (theorem proving system, **TPS**), такі як Otter [37], SPASS [38], або Vampire [39].

В режимі пошуку виведення САД приймає вхідні тексти, записані безпосередньо в мові першого порядку. Модулі [FOL] та [TPTP] обробляють, відповідно, завдання виведення у власному синтаксисі САД та в форматі бібліотеки TPTP [40].

Штрихові лінії на рисунку 1 відображають зв'язки, що мають бути реалізовані в системі САД в майбутньому: кооперація з зовнішніми системами комп'ютерної алгебри та автоматична трансляція доведень, знайдених системою в мову ForTheL.

Система САД реалізована в мовах Haskell [41] та C.

РОЗДІЛ 1
МОВА МАТЕМАТИЧНИХ ТЕКСТІВ FORTHEL

Мова ForTheL (Formal Theory Language) — це формальна мова для запису математичних текстів, що імітує природну англійську мову математичних публікацій. Існує ряд причин, за яких ми віддаємо перевагу такому "людському" способу подання, радше ніж стислій нотації деякої традиційної мови математичної логіки. Розглянемо дві з них.

По-перше, текст, що складається з речень людської мови, буде легшим для читача (особливо для читача, необізнаного у математичній логіці), ніж сукупність формул, побудованих з кванторів, скобок, логічних зв'язок та іншого. Отже першою причиною є — надати користувачеві системи зручне та комфортне робоче середовище.

По-друге, математичний текст, записаний природною мовою, містить багато інформації, яка лежить за межами мови логіки і, зазвичай, зникає при перекладі. В людському мовленні ми зустрічаємо іменники, що позначають індивідні сутності або класи індивідів; прикметники, дієприкметники та дієслова, що виступають в ролі атрибутів і обмежують класи; прикметники та дієслова, що діють як присудки і можуть вказувати на відношення між різними сутностями. В традиційному математичному тексті, ми зустрічаємо визначення та аксіоми, важливі теореми та допоміжні леми, різноманітні схеми міркування. Там, де людська мова розрізняє, мова математичної логіки уніфікує: іменники-класи, прикметники, дієслова перетворюються на предикатні символи; аксіоми, визначення, леми постають рівноправними формулами-антецедентами; доведення від супротивного, доведення за розглядом випадків, кроки розкриття визначень або застосування лем — трансформуються у кроки застосування *modus ponens*, або резолюції, або табличних правил.

На наш погляд, вибір проміжних тверджень на роль лем, вибір нових понять, які вводяться у визначеннях, вибір схем доведення, та інші засоби організації математичного тексту несуть важливу інформацію про цей текст і мають бути враховані разом з суто логічним змістом.

Математик-людина застосовує визначення інакше ніж аксіоми та леми, оперує класами інакше ніж відношеннями, тобто використовує всі ці дистинкції в процесі пошуку доведення. Отже, різноманітність людської мови не є надлишковою, вона має зберігтись в формальному поданні проблеми, і, за умови, що вдасться знайти адекватний шлях формалізації, це дасть можливість розширити дедуктивні потужності машини: запровадити евристичні процедури міркування або методи скорочення пошукового простіру, що спиратимуться на "поза-логічну" інформацію, узяту з формального, але деякою мірою природного вхідного тексту.

Мова ForTheL має спільні риси з такими формальними мовами, як Attempto Controlled English (ACE) [42] та Common Logic Controlled English (CLCE) [43]. Відмітністю ForTheL є розвинутий синтаксис для подання структурованих текстів, зокрема, математичних доведень.

Перший розділ дисертаційної роботи присвячено мові ForTheL, її синтаксису та семантиці. Опис йде в наступному порядку:

1. Лексична структура, що визначає, яким чином послідовність ASCII-символів перетворюється на ланцюг *лексем* ForTheL.

2. Наступний рівень синтаксису ForTheL: частини речення або *юніти*: терми, поняття, предикати, пропозиції. Юніти будуються з *синтаксичних примітивів* за правилами граматики ForTheL. Семантика пропозицій може бути виражена формулою першого порядку.

3. Далі ми розглядаємо *розділи* ForTheL. Найменшим розділом є *речення*. Сукупності речень формують складні розділи: нижнього рівня, такі як доведення або випадки, та верхнього рівня, такі як аксіоми, визначення, та теореми. ForTheL-розділи не мають фіксованої семантики, вони наділяються сукупністю характеристик, що визначаються їхнім змістом та контекстом.

4. Нарешті, *ForTheL-текст* є послідовністю розділів верхнього рівня та спеціальних конструкцій, *інтродукторів*, які вводять нові синтаксичні примітиви в сигнатуру текста.

1.1. Лексична структура

Ми використовуємо розширені форми Бекуса-Наура для запису продукцій граматики ForTheL. Нетермінальні символи записуються курсивом (напр. *attribute*), термінальні — шрифтом typewriter (напр. `every`). Продукції мають форму:

$$nonterm \;\rightarrow\; alt_1 \,|\, alt_2 \,|\, \ldots \,|\, alt_n$$

і ми застосовуємо наступну нотацію:

$pat_1 \,|\, pat_2$	вибір
$(\, pattern \,)$	групування
$[\, pattern \,]$	не більше одного входження
$\{\, pattern \,\}$	нуль або більше послідовних входжень

Лексична структура ForTheL визначається наступним чином:

$$\begin{aligned}
lexeme &\rightarrow word \mid symbol \\
word &\rightarrow alphanum \,\{\, alphanum \,\} \\
alphanum &\rightarrow alpha \mid numeric \mid \texttt{_} \\
alpha &\rightarrow small \mid capital \\
small &\rightarrow \texttt{a} \mid \ldots \mid \texttt{z} \\
capital &\rightarrow \texttt{A} \mid \ldots \mid \texttt{Z} \\
numeric &\rightarrow \texttt{0} \mid \ldots \mid \texttt{9} \\
symbol &\rightarrow \texttt{(} \mid \texttt{)} \mid \texttt{[} \mid \texttt{]} \mid \texttt{\{} \mid \texttt{\}} \mid \texttt{<} \mid \texttt{>} \mid \texttt{`} \mid \texttt{'} \mid \texttt{"} \mid \texttt{/} \mid \texttt{\textbackslash} \mid \texttt{|} \mid \texttt{\%} \\
&\quad \mid \texttt{!} \mid \texttt{?} \mid \texttt{@} \mid \texttt{\$} \mid \texttt{\&} \mid \texttt{\%} \mid \texttt{\~{}} \mid \texttt{+} \mid \texttt{-} \mid \texttt{*} \mid \texttt{=} \mid \texttt{:} \mid \texttt{;} \mid \texttt{,} \mid \texttt{.} \\
whitespace &\rightarrow whitetoken \,\{\, whitetoken \,\} \\
whitetoken &\rightarrow space \mid newline \mid comment \\
comment &\rightarrow \texttt{\#}\,\{\text{будь-який символ крім кінця рядка}\}\, newline \\
space &\rightarrow \text{пропуск} \mid \text{табуляція} \\
newline &\rightarrow \text{кінець рядка}
\end{aligned}$$

В процесі лексичного аналізу, за одну лексему приймається найдовша послідовність символів, що відповідає правилам, наведеним вище. Так,

зокрема, дві лексеми *word* не можуть стояти поруч — їх обов'язково має розділяти лексема *symbol* або *whitespace*. Лексеми *whitespace* (пропуски) служать для розмежування лексем і не згадуються в подальших правилах граматики ForTheL.

Змінні в ForTheL позначаються латинськими літерами.

$$variable \rightarrow alpha$$

Регістр літери враховується, тобто `x` та `X` є різними змінними.

1.2. Юніти ForTheL

Для того, щоб дати уяву про цей рівень мови ForTheL, наведемо декілька прикладів ForTheL-пропозицій. Для кожної пропозиції вказано її переклад в мову першого порядку.

ForTheL: `every bird is a feathery animal`

Переклад: $\forall\, x\, (\mathrm{aBird}(x) \supset (\mathrm{aAnimal}(x) \wedge \mathrm{isFeathery}(x)))$

ForTheL: `every subset of some set S is equal to S`

Переклад: $\exists\, S\, (\mathrm{aSet}(S) \wedge \forall\, x\, (\mathrm{aSubsetOf}(x, S) \supset x = S))$

ForTheL: `every natural number m greater than 0 divides m!`

Переклад: $\forall\, m\, ((\mathrm{aNumber}(m) \wedge \mathrm{isNatural}(m) \wedge$
$\wedge\, \mathrm{isGreaterThan}(m, \mathrm{Zero})) \supset \mathrm{Divides}(m, \mathrm{Factorial}(m)))$

ForTheL: `if X and Y are close to Z then X and Y are close`

Переклад: $(\mathrm{isCloseTo}(X, Z) \wedge \mathrm{isCloseTo}(Y, Z)) \supset \mathrm{isCloseTo}(X, Y)$

ForTheL: `no equation in G has a positive solution`

Переклад: $\forall\, x\, ((\mathrm{aEquation}(x) \wedge \mathrm{isIn}(x, G)) \supset$
$\supset \neg\exists\, y\, (\mathrm{aSolutionOf}(y, x) \wedge \mathrm{isPositive}(y)))$

Подібно до того, як ForTheL-текст є композицією речень та складних розділів, ForTheL-речення є композицією юнітів. Юніти розділяються на декілька типів:

- *поняття* позначає клас індивідних об'єктів, можливо параметризований: `real number`, `element of S`, `series that converges to N`.
- *терм* позначає індивідний об'єкт, або вказуючи деяке конкретне значення: `N`, `the complement of S`, `X * Y`; або за допомогою квантора по об'єму класу, визначеного деяким поняттям: `every set`, `some divisor of M`.
- *предикат* позначає властивість індивідного об'єкту: `empty`, `divides N`, `is a subset of S`. Застосовуючи предикат до терма, отримаємо пропозицію; застосовучи предикат до поняття як атрибут, отримаємо нове поняття, з обмеженим класом.
- *пропозиція* позначає вислів, який може бути істинним або хибним: `X > Y`, `every divisor of N is a natural number`, `there exists a countable set`.

Третя пропозиція з попереднього прикладу розкладається на юніти наступним чином:

Як і будь-яка природна мова, мова ForTheL не є вільною від неоднозначності. Синтаксис ForTheL дозволяє записувати юніти, що мають більш, ніж один сенс. Наприклад, в пропозиції:

```
some point of any straight line that crosses L lies on L
```

предикат `crosses L` може бути віднесений як до поняття `point`, так і до поняття `line`. Не маючи можливості звертатися до дедуктивного апарату системи, парсер не в змозі визначити правильний варіант, тому подібні конструкції відкидаються. Як правило, нескладно знайти однозначне переформулювання:

```
every straight line that crosses L has a point that lies on L
```

або застосувати дужки:

```
some point of (any straight line that crosses L) lies on L
```

1.2.1. Синтаксичні примітиви

Синтаксичні примітиви є цеглинками, з яких будуються юніти. В кожен момент синтаксичного аналізу тексту, ми маємо справу з певним набором примітивів, що відповідає сигнатурі тієї частини тексту, що вже прочитана парсером. Ми називаємо цей набір *поточним словником*, підкреслюючи, по-перше, що цей набір притаманний тексту, що розглядається, а, по-друге, що він є динамічним. В ForTheL є шість груп так званих *базових примітивів*:

- *загальні іменники*, що утворюють атомарні поняття:
  `natural number`, `element of` arg_1, `function from` arg_1 `to` arg_2
- *визначені іменники*, що утворюють функціональні терми:
  `zero`, `power set of` arg_1, `sum of` arg_1 `and` arg_2
- *прикметники* та *дієслова*, що утворюють предикати:
  `prime`, `converges`, `equal to` arg_1
- *функціональні символи*, що утворюють символьні терми:
  arg_1 `+` arg_2, `min` arg_1, `exp(`arg_1`)`
- *предикатні символи*, що утворюють символьні пропозиції:
  arg_1 `<=` arg_2, arg_1 `:` arg_2 `->` arg_3

Базові примітиви вводяться безпосередньо в ForTheL-тексті за допомогою службових конструкцій, що називаються *інтродукторами*. Додаткові групи примітивів автоматично породжуються за нововведеними базовими примітивами. Так, наприклад, примітиви-іменники з аргументним місцем після прийменника `of`, такі як `subset of` arg_1 або `complement of` arg_1 `to` arg_2 породжують додаткові примітиви, що вживатимуться в предикатах виду: `of an infinite cardinality`, `has an element`. Породжувані групи примітивів розглядаються в наступних підрозділах.

Синтаксис інтродукторів визначається наступним чином (один рядок для кожної базової групи):

$$
\begin{aligned}
\mathit{introductor} \rightarrow\ & \texttt{[}\ (\ \texttt{a} \mid \texttt{an}\ )\ \mathit{pattern}\ [\ \texttt{@}\ [\ \texttt{a} \mid \texttt{an}\ ]\ \mathit{nounNotion}\ ]\ \texttt{]} \\
\mid\ & \texttt{[the}\ \mathit{pattern}\ [\ \texttt{@}\ \mathit{plainTerm}\ ]\ \texttt{]} \\
\mid\ & \texttt{[}\ \mathit{variable}\ \texttt{is}\ \mathit{pattern}\ [\ \texttt{@}\ \mathit{proposition}\ ]\ \texttt{]} \\
\mid\ & \texttt{[}\ \mathit{variable}\ \mathit{pattern}\ [\ \texttt{@}\ \mathit{proposition}\ ]\ \texttt{]} \\
\mid\ & \texttt{[}\ \mathit{symbPattern}\ \texttt{@}\ \mathit{plainTerm}\ \texttt{]} \\
\mid\ & \texttt{[}\ \mathit{symbPattern}\ \texttt{@}\ \mathit{proposition}\ \texttt{]}
\end{aligned}
$$

Кожний інтродуктор починається зі *зразка*, який визначає синтаксис нововведеного примітиву. Примітив може бути (а символьний примітив — має бути) введений як *синонім* для деякого юніту відповідного типу; в цьому випадку, після зразка ставиться символ `@`, за яким йде *означуваний* юніт. Нетермінальні символи *nounNotion*, *plainTerm* та *proposition* будуть визначені нижче. Всі примітиви, що зустрічаються в означуваних юнітах, мають бути присутні в поточному словнику; таким чином, самопосилання або взаємна синонімія не дозволяються.

Зразок є непустим ланцюгом *токенів* (термінальних символів нового примітиву) та змінних (що вказують аргументні місця):

$$
\begin{aligned}
\mathit{pattern} &\rightarrow \{\, \mathit{tokens}\,\}\ \mathit{tokens}\ [\, \mathit{variable}\ \{\, \mathit{tokens}\ \mathit{variable}\,\}\,] \\
\mathit{tokens} &\rightarrow \mathit{token}\ [\, \texttt{/}\ \mathit{token}\,] \\
\mathit{token} &\rightarrow \mathit{small}\ \{\, \mathit{small}\,\} \\
\mathit{symbPattern} &\rightarrow [\, \mathit{variable}\,]\ \mathit{symbToken} \\
&\qquad\qquad \{\, \mathit{variable}\ \mathit{symbToken}\,\}\ [\, \mathit{variable}\,] \\
&\mid\ \mathit{word}\ \texttt{(}\ \mathit{variable}\ \{\ \texttt{,}\ \mathit{variable}\,\}\ \texttt{)} \\
&\mid\ \mathit{word}\ [\, \mathit{variable}\,] \\
\mathit{symbToken} &\rightarrow \mathit{symbol}\ \{\, \mathit{symbol}\,\}
\end{aligned}
$$

В зразках для несимвольних примітивів дозволяється перелічувати декілька варіантів одного токену. Таким чином, іменникі та дієслова можуть вживатися як в однині, так і в множині. Токени не повинні містити пропусків. Артиклі та форми дієслова “to be” не можуть бути токенами.

В інтродукторах дієслів головна змінна не повинна називатися `a` або `A`, щоб уникнути коллізії з інтродукцією загальних іменників. Всі змінні в зразку мають бути різні. Якщо інтродуктор вводить синонім, всі вільні змінні (1.2.6) в означуваному юніті повинні входити в зразок.

Парсер може завжди визначити, який тип примітива вводиться в інтродукторі — або за формою зразка, або, якщо вводиться символьний примітив, розпізнавши означуваний юніт як терм або пропозицію.

Слідом за обробкою інтродуктора, до граматики додаються нові продукції для відповідних груп примітивів:

Інтродуктор	Примітив
`[a subset/subsets of S]`	(`subset` | `subsets`) [*names*] (`of`) *term*
`[the root/roots of S]`	(`root` | `roots`) (`of`) *term*
`[U is infinite @ U is not finite]`	(`infinite`)
`[m divides/divide n]`	(`divides` | `divide`) *term*
`[Pow x @ the power set of x]`	(`Pow`) *symbTerm*
`[x >= y @ x is not less than y]`	*symbTerms* (`>=`) *symbTerm*

Токени зразку перетворюються на групи вибору термінальних символів, а змінні замінюються на нетермінальні символи *term* (*symbTerm*, для символьних примітивів), що “считуватимуть” аргументи примітиву. Якщо зразок предикатного символа починається зі змінної, вона замінюється на нетермінальний символ *symbTerms*. Присутність або відсутність означуваного юніту не впливає на синтаксис нового примітива. В загальних іменниках, опціональний нетермінал *names* (дивись 1.2.2) додається за один токен до першого аргументного місця, або в кінці зразку, якщо аргументів немає.

Регістр враховується для токенів символьних примітивів та для змінних. Таким чином, `subset of S`, `Subset of S`, та `sUbSeT oF S` означа-

ють одне й те саме поняття, але `sin x`, `Sin x`, `sin X` означають три різних терма.

Правила граматики, що відповідають за розпізнавання примітивів, є єдиними правилами, що змінюються в процесі аналізу тексту. В наступних підрозділах ці правила будуть показуватись як гіпотетичні "стопкадри" в деякі довільні моменти обробки тексту.

1.2.2. Поняття

Будь-який юніт поняття будується з *атомарного поняття*, до якого додаються *атрибути*. Два типа синтаксичних примітивів використовуються для утворення атомарних понять:

$$
\begin{aligned}
primaryNotion &\rightarrow primClassNoun \mid notionSymbol \\
primClassNoun &\rightarrow (\,\texttt{set} \mid \texttt{sets}\,)\ [\,names\,] \\
&\mid\ (\,\texttt{element} \mid \texttt{elements}\,)\ [\,names\,]\ (\,\texttt{of}\,)\ term \\
&\mid\ \ldots \\
notionSymbol &\rightarrow primNotionSymbol \mid (\ primNotionSymbol\ ) \\
primNotionSymbol &\rightarrow names\ (\,\texttt{<<}\,)\ symbTerm \\
&\mid\ names\ (\,\texttt{:}\,)\ symbTerm\ (\,\texttt{->}\,)\ symbTerm \\
&\mid\ \ldots \\
names &\rightarrow variable\ \{\ ,\ variable\,\}
\end{aligned}
$$

Символи понять є породжуваними примітивами. Вони породжуються автоматично з дескриптивних предикатних символів (дивись 1.2.7). На перше аргументне місце в символі поняття ставиться нетермінальний символ *names*, решта залишається такою, як в предикатному символі. Символьні поняття, що наведені вище, могли бути породжені за такими інтродукторами:

```
[x << y @ x is an element of y]
[f : D -> R @ f is a function from D to R]
```

Кожний юніт поняття характеризується списком *імен*. Ці ідентифікатори звертаються до окремих об'єктів з класу та відіграють ту ж саму роль, що імена змінних поруч з квантором. Ми вже бачили імена в пропозиції:

```
every natural number m greater than 0 divides m!
```

ForTheL також дозволяє поняття з декількома іменами:

```
for all real numbers X,Y (X*Y) is a real number
```

Вживання імен не є обов'язковим:

```
every even natural number greater than 2 is compound

L,M are parallel straight lines
```

Атрибути використовуються для накладення обмежень на клас, означений поняттям. З точки зору синтаксису, атрибути базуються на предикатах та пропозиціях ForTheL:

$$leftAttribute \rightarrow primSimpleAdjective \mid primSimpleAdjectiveM$$

$$\begin{aligned} rightAttribute \rightarrow\ & isPredicate\ \{\texttt{and}\ isPredicate\} \\ \mid\ & \texttt{that}\ doesPredicate\ \{\texttt{and}\ doesPredicate\} \\ \mid\ & \texttt{such that}\ proposition \end{aligned}$$

Прості прикметники, що утворюють ліві атрібути, також є породжуваними примітивами. Вони формуються з тих прикметників та мультиприкметників (див. 1.2.4), які не мають аргументних місць:

$$primSimpleAdjective \rightarrow (\texttt{prime}) \mid (\texttt{empty}) \mid (\texttt{natural}) \mid \ldots$$

$$primSimpleAdjectiveM \rightarrow (\texttt{equal}) \mid (\texttt{parallel}) \mid (\texttt{disjoint}) \mid \ldots$$

Визначимо тепер синтаксис поняття:

$$notion \rightarrow \{leftAttribute\}\ primaryNotion\ [rightAttribute]$$

Наступні конструкції є прикладами правильно побудованих понять:

```
cyclic group G
injective f : Nat -> Nat that maps 0 to 0
real number greater than 0 and less than 1
natural numbers q,r such that n = (q * m) + r and r < m
```

Відмітимо, що останній юніт насправді описує два різних класа: один для `q` і один для `r`.

1.2.3. Терми

В мові ForTheL існує два типа термів: визначені терми та квантифіковані поняття:

$$term \rightarrow quantifiedNotion \mid definiteTerm$$

Квантифіковані поняття дозволяють формулювати універсальні або екзистенційні твердження про індивідні об'єкти з класу:

$$\begin{aligned} quantifiedNotion &\rightarrow realQuantifiedNotion \\ &\mid \texttt{(}\ realQuantifiedNotion\ \texttt{)} \\ \\ realQuantifiedNotion &\rightarrow (\texttt{every} \mid \texttt{each} \mid \texttt{all} \mid \texttt{any})\ notion \\ &\mid \texttt{some}\ notion \\ &\mid \texttt{no}\ notion \end{aligned}$$

Зауваження. Квантифіковані поняття, що стоять в аргументних місцях синтаксичних примітивів не можуть мати більше одного імені. Так, юніти `N divides some numbers X,Y,Z`, `a subset of finite sets A,B`, `the orders of groups G,H` не є правильно побудованими.

Визначений терм або є змінною або утворюється застосуванням функції до термів-аргументів. ForTheL дозволяє записувати такі застосування за допомогою англійських слів або функціональних символів. В граматиці прийнято наступні правила щодо приоритету та напрямку асоціювання функціональних символів.:

- *постфіксні* символи, чиї примітиви завершуються токеном, мають найвищий приоритет;
- *префіксні* символи, чиї примітиви починаються з токена, а завершуються аргументним місцем, мають менший приоритет, ніж постфіксні символи;
- *інфіксні* символи, чиї примітиви починаються і завершуються аргументними місцями, мають найнижчий приоритет серед функціональних символів і ассоціюються зліва направо.

Для перевпорядкування символьного виразу можна вживати дужки.

$$\begin{aligned}
definiteTerm &\rightarrow realDefiniteTerm \mid \texttt{(}\ realDefiniteTerm\ \texttt{)} \\
realDefiniteTerm &\rightarrow [\,\texttt{the}\,]\ primDefiniteNoun \mid symbTerm \\
symbTerm &\rightarrow primInfixFunctionSymbol \mid prefixSymbTerm \\
prefixSymbTerm &\rightarrow primPrefixFunctionSymbol \mid postfixSymbTerm \\
postfixSymbTerm &\rightarrow primPostfixFunctionSymbol \mid \texttt{(}\ symbTerm\ \texttt{)} \\
&\mid\ variable
\end{aligned}$$

Наведемо типові примітиви для визначених іменників та функціональних символів:

$$\begin{aligned}
primDefiniteNoun &\rightarrow (\,\texttt{zero} \mid \texttt{zeroes}\,) \\
&\mid\ (\,\texttt{order} \mid \texttt{orders}\,)\ (\,\texttt{of}\,)\ term \\
&\mid\ \ldots \\
primInfixFunctionSymbol &\rightarrow prefixSymbTerm\ (\,\texttt{*}\,)\ symbTerm \\
&\mid\ \ldots \\
primPrefixFunctionSymbol &\rightarrow (\,\texttt{min}\,)\ prefixSymbTerm \\
&\mid\ \ldots
\end{aligned}$$

$$
\begin{aligned}
primPostfixFunctionSymbol \;\rightarrow\; & \texttt{(}\;0\;\texttt{)} \\
\mid\; & \texttt{(}\;\texttt{exp}\;\texttt{)}\;\texttt{(}\;\texttt{(}\;\texttt{)}\;symbTerm\;\texttt{(}\;\texttt{)}\;\texttt{)} \\
\mid\; & postfixSymbTerm\;\texttt{(}\;\texttt{(}\;\texttt{)}\;symbTerm\;\texttt{(}\;\texttt{)}\;\texttt{)} \\
\mid\; & \ldots
\end{aligned}
$$

Звернемо увагу на останній примітив в групі постфіксних функціональних символів. Він вводить операцію застосування функції до аргумента як абстрактну бінарну операцію, використовуючи такий самий синтаксис, що й сигнатурні функції, наприклад, `exp`.

Простий терм — це терм, що не містить квантифікованих понять. Інакше кажучи, простий терм — це визначений терм, чиї аргументи, якщо вони в нього є, є простими термами:

$$
\begin{aligned}
plainTerm \;\rightarrow\; & realPlainTerm \mid \texttt{(}\; realPlainTerm \;\texttt{)} \\
realPlainTerm \;\rightarrow\; & [\,\texttt{the}\,]\; primPlainNoun \mid symbTerm \\
primPlainNoun \;\rightarrow\; & \texttt{(}\;\texttt{zero}\,|\,\texttt{zeroes}\;\texttt{)} \\
\mid\; & \texttt{(}\;\texttt{order}\,|\,\texttt{orders}\;\texttt{)}\;\texttt{(}\;\texttt{of}\;\texttt{)}\; plainTerm \\
\mid\; & \ldots
\end{aligned}
$$

Кожен визначений іменник автоматично породжує примітив-близнюк з групи $primPlainNoun$, де в аргументних місцях стоїть нетермінал $plainTerm$ замість $term$.

1.2.4. Предикати

Функції та поняття описують об'єкти; предикати виражають властивості об'єктів та відношення між ними. В ForTheL існує три типа атомарних предикатів: предикати, що будуються з примітивів прикметників та дієслів; предикати, що стверджують приналежність до класу, означеного поняттям ("is a"-предикати); предикати, що виражають існування певного об'єкту, пов'язаного з суб'єктом предиката ("has"-predicates). Первинні

предикати можуть заперечуватись та об'єднуватись в кон'юнкції:

$$
\begin{aligned}
\mathit{doesPredicate} \rightarrow\ & [\,\texttt{does} \mid \texttt{do}\,]\ [\,\texttt{not}\,]\ \mathit{primVerb} \\
\mid\ & [\,\texttt{does} \mid \texttt{do}\,]\ [\,\texttt{not}\,]\ [\,\texttt{pairwise}\,]\ \mathit{primVerbM} \\
\mid\ & (\,\texttt{has} \mid \texttt{have}\,)\ \mathit{hasPredicate} \\
\mid\ & (\,\texttt{is} \mid \texttt{are} \mid \texttt{be}\,)\ \mathit{isPredicate}\ \{\,\texttt{and}\ \mathit{isPredicate}\,\} \\
\mid\ & (\,\texttt{is} \mid \texttt{are} \mid \texttt{be}\,)\ \mathit{is\_aPredicate}\ \{\,\texttt{and}\ \mathit{is\_aPredicate}\,\}
\end{aligned}
$$

$$
\begin{aligned}
\mathit{isPredicate} \rightarrow\ & [\,\texttt{not}\,]\ \mathit{primAdjective} \\
\mid\ & [\,\texttt{not}\,]\ [\,\texttt{pairwise}\,]\ \mathit{primAdjectiveM} \\
\mid\ & (\,\texttt{with} \mid \texttt{of} \mid \texttt{having}\,)\ \mathit{hasPredicate}
\end{aligned}
$$

$$
\begin{aligned}
\mathit{is\_aPredicate} \rightarrow\ & [\,\texttt{not}\,]\ [\,\texttt{a} \mid \texttt{an}\,]\ \mathit{nounNotion} \\
\mid\ & [\,\texttt{not}\,]\ \mathit{definiteTerm}
\end{aligned}
$$

$$
\begin{aligned}
\mathit{hasPredicate} \rightarrow\ & [\,\texttt{a} \mid \texttt{an} \mid \texttt{the}\,]\ \mathit{possessed}\ \{\,\texttt{and}\ [\,\texttt{a} \mid \texttt{an} \mid \texttt{the}\,]\ \mathit{possessed}\,\} \\
\mid\ & \texttt{no}\ \mathit{possessed}
\end{aligned}
$$

Примітиви прикметників та дієслів виглядають наступним чином:

$$
\begin{aligned}
\mathit{primVerb} \rightarrow\ & (\,\texttt{converges} \mid \texttt{converge}\,) \\
\mid\ & (\,\texttt{divides} \mid \texttt{divide}\,)\ \mathit{term} \\
\mid\ & (\,\texttt{belongs} \mid \texttt{belong}\,)\ (\,\texttt{to}\,)\ \mathit{term} \\
\mid\ & (\,\texttt{joins} \mid \texttt{join}\,)\ \mathit{term}\ (\,\texttt{with}\,)\ \mathit{term} \\
\mid\ & \ldots
\end{aligned}
$$

$$
\begin{aligned}
\mathit{primAdjective} \rightarrow\ & (\,\texttt{prime}\,) \\
\mid\ & (\,\texttt{dividing}\,)\ \mathit{term} \\
\mid\ & (\,\texttt{equal}\,)\ (\,\texttt{to}\,)\ \mathit{term} \\
\mid\ & (\,\texttt{less}\,)\ (\,\texttt{than}\,)\ \mathit{term} \\
\mid\ & \ldots
\end{aligned}
$$

Примітив `equal to` не треба вводити за допомогою інтродуктора, оскільки він є передвизначеним в граматиці мови. Зауважимо, що ми можемо

вводити дієприкметники як звичайні прикметники. Це має робитися явно, за допомогою окремого інтродуктора.

Примітиви дієслів та прикметників можуть автоматично породжувати *мультиприміmиви* за наступним правилом. Візьмемо примітив з груп $primVerb$ або $primAdjective$, який має принаймні один аргумент. Якщо перший аргумент в цьому примітиві йде після прийменника, і за ним немає сполучника (`and`), то породжується новий примітив в групі $primVerbM$ або, відповідно, $primAdjectiveM$, шляхом видалення першого аргументного місця та передуючого йому токена-прийменника.

Так, примітив прикметника ((`parallel`) (`to`) $term$) породжує мультиприкметник ((`parallel`)), що додається разом до $primAdjectiveM$ та до $primSimpleAdjectiveM$; а примітив дієслова ((`commutes` | `commute`) (`with`) $term$ (`wrt`) $term$) породжує мультидієслово ((`commutes`|`commute`) (`wrt`) $term$).

Предикати, що використовують мультиприміmиви (*мультипредикати*) мають застосовуватись до декількох об'єктів: `X,Y,Z commute wrt N`, `parallel lines l,m`. Хоча подібні вирази мають сенс лише для симетричних відношень, система не в змозі відслідковувати на синтаксичному рівні, чи володіє тий чи інший примітив цією властивістю, отже нові мультиприміmиви породжуються для всіх примітивів прикметників та дієслів.

За замовченням вважається, що мультиприміmиви описують симетричні та транзитивні відношення. Так, пропозиція `A,B,C are equal` буде переведена в формулу `A is equal to B and B is equal to C`. У випадку нетранзитивних відношень необхідно вживати опціональний прислівник `pairwise`. Тоді пропозиція `A,B,C are pairwise disjoint` буде коректно перекладена в `A is disjoint with B and A is disjoint with C and B is disjoint with C`.

$$
\begin{aligned}
primVerbM \;\rightarrow\; & (\,\texttt{collides} \,|\, \texttt{collide}\,) \\
| \;\; & (\,\texttt{commutes} \,|\, \texttt{commute}\,)\;(\,\texttt{wrt}\,)\;term \\
| \;\; & \ldots
\end{aligned}
$$

$$
\begin{aligned}
primAdjectiveM \;\rightarrow\; & (\,\texttt{equal}\,) \\
\mid\; & (\,\texttt{adjacent}\,)\;(\,\texttt{in}\,)\;term \\
\mid\; & \ldots
\end{aligned}
$$

За допомогою “is a”-предикатів ми виражаємо, що об’єкт або належить до класу деякого поняття або дорівнює деякому визначеному терму: `X is a natural number`, `X is the order of G`. В цих предикатах не можна вживати символьні поняття, тому ми вводимо спеціальний нетермінал $nounNotion$ для понять, що формуються з загальних іменників.

$$nounNotion \;\rightarrow\; \{\, leftAttribute \,\}\; primClassNoun\; [\, rightAttribute \,]$$

Зауваження. Загальні іменники в $nounNotion$ повинні мати щонайбільше одне ім’я. Якщо такий іменник має ім’я або використовується в означуваному понятті в інтродукторі, він не повинен мати в атрибутах мультипредикатів. Так, юніти `an equal number X`, `a line N that is parallel`, `a set S such that S is disjoint` є неправильно побудованими.

В “has”-предикатах також використовуються породжувані примітиви, що утворюються з загальних та визначених іменників:

$$
\begin{aligned}
possessed \;\rightarrow\; & \{\, leftAttribute \,\}\; primPossessedNoun \\
& [\, rightAttribute \,]
\end{aligned}
$$

$$
\begin{aligned}
primPossessedNoun \;\rightarrow\; & (\,\texttt{element} \mid \texttt{elements}\,)\;[\, names \,] \\
\mid\; & (\,\texttt{solution} \mid \texttt{solutions}\,)\;[\, names \,] \\
\mid\; & (\,\texttt{order} \mid \texttt{orders}\,)\;[\, names \,] \\
\mid\; & \ldots
\end{aligned}
$$

Ці примітиви породжуються за наступним правилом. Візьмемо примітив, що має принаймні один аргумент, з груп $primClassNoun$ або $primDefiniteNoun$. Якщо перше аргументне місце в цьому примітиві слідує за прийменником (`of`), а за ним немає сполучника (`and`), то породжується новий примітив притяжного іменника шляхом вилучення першого аргументного місця та передуючого йому токена (`of`). Якщо

початковий примітив був визначеним іменником, то в продукцію також додається опціональна група [*names*] .

Так, примітив загального іменника

(`ambassador` | `ambassadors`) [*names*] (`of`) *term* (`in`) *term*

породжує новий примітив

(`ambassador` | `ambassadors`) [*names*] (`in`) *term*

але примітив `union of _ and _` не породжує нового примітива.

Наведемо декілька прикладів пропозицій з "has"-предикатами:

```
X has no elements

every set of a finite cardinality is finite

F has the domain D and the range R such that D [= R
```

В другій пропозиції "has"-предикат є правим атрибутом поняття.

1.2.5. Пропозиції

Пропозиції в ForTheL відповідають формулам мови математичної логіки. По-перше, розглянемо чотири групи *атомарних* пропозицій:

$$\begin{aligned} \mathit{atomicProposition} \;\rightarrow\; & \mathit{simpleProposition} \\ \mid\; & \mathit{thereIsProposition} \\ \mid\; & [\,\texttt{we have}\,]\ \mathit{symbProposition} \\ \mid\; & [\,\texttt{we have}\,]\ \mathit{specialProposition} \end{aligned}$$

Прості пропозиції утворюються застосуванням предикатів до термів:

$$\mathit{simpleProposition} \;\rightarrow\; \mathit{terms}\ \mathit{doesPredicate}\ \{\,\texttt{and}\ \mathit{doesPredicate}\,\}$$

$$\mathit{terms} \;\rightarrow\; \mathit{term}\ \{\,(\ \texttt{,}\ \mid \texttt{and}\ )\ \mathit{term}\,\}$$

Так звані “there is”-пропозиції стверджують або заперечують непустоту класу деякого поняття:

$$\begin{aligned} \mathit{thereIsProposition} \;\rightarrow\; & \texttt{there}\ (\ \texttt{exists} \,|\, \texttt{exist}\ )\ \mathit{notions} \\ |\;\; & \texttt{there}\ (\ \texttt{exists} \,|\, \texttt{exist}\ )\ \texttt{no}\ \mathit{notion} \end{aligned}$$

$$\mathit{notions} \;\rightarrow\; [\ \texttt{a} \,|\, \texttt{an}\ ]\ \mathit{notion}\ \{\ (\ \texttt{,} \,|\, \texttt{and}\ )\ [\ \texttt{a} \,|\, \texttt{an}\ ]\ \mathit{notion}\ \}$$

Символьні пропозиції записуються в традиційному синтаксисі першого порядку, вони утворюються з символьних предикатів та термів. Звичайні пропозиції, взяті в дужки, також дозволяються всередині символьних пропозицій. В синтаксисі ForTheL, `<=>` означає еквівалентність, `=>` — імплікацію, `\/` —диз’юнкцію, а `/\` — кон’юнкцію. В наступному правилі для *symbProposition*, продукції впорядковані за зростанням пріоритету.

$$\begin{aligned} \mathit{symbProposition} \;\rightarrow\; & \texttt{forall}\ \mathit{notionSymbol}\ \mathit{symbProposition} \\ |\;\; & \texttt{exists}\ \mathit{notionSymbol}\ \mathit{symbProposition} \\ |\;\; & \mathit{symbProposition}\ \texttt{<=>}\ \mathit{symbProposition} \\ |\;\; & \mathit{symbProposition}\ \texttt{=>}\ \mathit{symbProposition} \\ |\;\; & \mathit{symbProposition}\ \texttt{\textbackslash/}\ \mathit{symbProposition} \\ |\;\; & \mathit{symbProposition}\ \texttt{/\textbackslash}\ \mathit{symbProposition} \\ |\;\; & \texttt{not}\ \mathit{symbProposition} \\ |\;\; & \texttt{(}\ \mathit{proposition}\ \texttt{)} \\ |\;\; & \mathit{primPredicateSymbol} \end{aligned}$$

$$\begin{aligned} \mathit{primPredicateSymbol} \;\rightarrow\; & \mathit{symbTerms}\ (\ \texttt{=}\ )\ \mathit{symbTerm} \\ |\;\; & \mathit{symbTerms}\ (\ \texttt{!=}\ )\ \mathit{symbTerm} \\ |\;\; & \mathit{symbTerms}\ (\ \texttt{-<-}\ )\ \mathit{symbTerm} \\ |\;\; & \mathit{symbTerms}\ (\ \texttt{:}\ )\ \mathit{symbTerm}\ (\ \texttt{->}\ )\ \mathit{symbTerm} \\ |\;\; & \ldots \end{aligned}$$

$$\mathit{symbTerms} \;\rightarrow\; \mathit{symbTerm}\ \{\ \texttt{,}\ \mathit{symbTerm}\ \}$$

Бінарні предикатні символи `=`, `!=` та `-<-` є передвизначеними в мові ForTheL. Перші два є синонімами для відношення (не)рівності, а третє позначає абстрактне фундоване відношення, що використовується для доведень за індукцією (див. 1.3.2).

Спеціальні пропозиції `thesis` та `contrary` позначають поточну цільову пропозицію та її заперечення (див. 1.3.4), а `contradiction` позначає хибність.

$$\begin{aligned} \mathit{specialProposition} \rightarrow\ & [\,\texttt{the}\,]\ \texttt{thesis} \mid [\,\texttt{the}\,]\ \texttt{contrary} \\ \mid\ & [\,\texttt{a} \mid \texttt{an}\,]\ \texttt{contradiction} \end{aligned}$$

Атомарні пропозиції комбінуються за допомогою сполучників:

$$\mathit{proposition} \rightarrow \mathit{headProposition} \mid \mathit{chainProposition}$$

$$\begin{aligned} \mathit{headProposition} \rightarrow\ & \texttt{for}\ \mathit{quantifiedNotion} \\ & \qquad \{\,\texttt{and}\ \mathit{quantifiedNotion}\,\}\ \mathit{proposition} \\ \mid\ & \texttt{if}\ \mathit{proposition}\ \texttt{then}\ \mathit{proposition} \\ \mid\ & \texttt{it is wrong that}\ \mathit{proposition} \end{aligned}$$

$$\begin{aligned} \mathit{chainProposition} \rightarrow\ & \mathit{andChain}\ [\,\texttt{and}\ \mathit{headProposition}\,] \\ \mid\ & \mathit{orChain}\ [\,\texttt{or}\ \mathit{headProposition}\,] \\ \mid\ & (\,\mathit{andChain} \mid \mathit{orChain}\,)\ \texttt{iff}\ \mathit{proposition} \end{aligned}$$

$$\mathit{andChain} \rightarrow \mathit{atomicProposition}\ \{\,\texttt{and}\ \mathit{atomicProposition}\,\}$$

$$\mathit{orChain} \rightarrow \mathit{atomicProposition}\ \{\,\texttt{or}\ \mathit{atomicProposition}\,\}$$

1.2.6. Формульний образ пропозицій

В процесі аналізу послідовності лексем, система розпізнає частини цієї послідовності як приклади певних нетерміналів відповідно до правил граматика. Будемо називати ці підпослідовності *входженнями* відповідних нетерміналів в текст. Зауважимо, що контекст має братися при цьому до уваги. Розглянемо дві пропозиції:

`some natural number N divides 1`

`for some natural number N (N divides 1)`

Строка `some natural number N` є входженням нетерміналу *term* в першій пропозиції, але не в другій. Також строка `natural number` не є входженням *notion* в обох твердженнях (хоча сама по собі вона є правильно побудованим поняттям), оскільки в ці входження має потрапити ім'я `N`.

Використовуючи відповідність між строками лексем та нетерміналами, ми перетворюємо пропозицію S в деяку формулу першого порядку $|S|$, яку називаємо *формульним образом* S. Для цього ми послідовно застосовуємо переписуючі правила, що перелічуються далі. Кожне правило має бути застосовано стільки раз, скільки можливо, перед тим як наступне правило буде застосовано в перший раз.

1. Видалення "синтаксичного сахару". В першу чергу необхідно дещо нормалізовати синтаксис для спрощення подальших переписуючих правил:

   (a) В усіх входженнях *primClassNoun* або *primPossessedNoun*, де під'юніт [*names*] є пустим (тобто не вказано жодного імені), ставимо одну нову змінну на відповідне місце.

   (b) Розбиваємо ланцюгі понять в кванторах над пропозиціями у входженнях *headProposition*:

   $$\texttt{for } (quantifiedNotion)_O \texttt{ and } (quantifiedNotion \{ \texttt{and } quantifiedNotion \})_T \ (proposition)_S$$
   $$\Rightarrow \quad \texttt{for } O \texttt{ for } T \ \ S$$

   (c) Видаляємо артиклі `a`, `an`, `the` та префікси `we have`. Заперечення `it is wrong that` замінюється на `not`. Дієслова `be`, `are`, `do`, `have`, `exist` переписуються у форму `is`. Кванторні слова `all`, `each`, `any` замінюються на `every`. У входженнях нетерміналів *terms* та *notions*, сполучники `and` замінюються комами.

2. Перетворення квантіфікованих понять I. В подальшому, ми позначаємо через **N** групу нетерміналів *primClassNoun*, *primPossessedNoun*

або $primNotionSymbol$[1]. Для кожного входження O нетерміналу з $\mathbf{N}$, позначаємо через $\mathbf{n}(O)$ список змінних, що перелічені в під'юніті $[names]$ відповідного примітиву. Ця нотація тривіально поширюється на нетермінали $notion$, $nounNotion$, $notionSymbol$, $quantifiedNotion$ та $possessed$. Після застосування правила (1a), список імен є непустим в усіх входженнях нетерміналів з $\mathbf{N}$.

Наступні правила видаляють квантифіковані поняття з входжень нетерміналу $term$ в підметах простих пропозицій та під кванторами. Грубо кажучі, кожне квантифіковане поняття O заміщується на список імен $\mathbf{n}(O)$ і відповідний квантор ставиться над пропозицією.

Розглянемо два входження S та O. Назвемо O *власним входженням* в S, якщо O міститься строго всередині S. Назвемо O *нативним входженням* в S, якщо O є власним в S, але не є власним в жодному входженні нетерміналу $rightAttribute$ в S.

Наприклад, входження X, Y та U нетерміналу $quantifiedNotion$ є нативними в наступній простій пропозиції, тоді як входження V не є нативним:

$$(\texttt{every member M of (some committee C)}_Y)_X \texttt{ declares}$$
$$(\texttt{some view O such that (every member N of C)}_V \texttt{ supports O})_U$$

Наступні переписуючі правила застосовуються для входжень нетерміналів $simpleProposition$ та $headProposition$. В антецеденті правила O є першим зліва нативним входження $quantifiedNotion$ в S. В консеквенті, O замінюється в S на $\mathbf{n}(O)$:

(a) $(terms\lceil(quantifiedNotion)_O\rfloor)_S$
$\qquad\qquad (doesPredicate\ \{\texttt{ and }\ doesPredicate\})_T$
$\Rightarrow\quad \texttt{for } O\ \ S\lceil O \to \mathbf{n}(O)\rfloor\ \ T$

(b) $\texttt{for } (quantifiedNotion\lceil(quantifiedNotion)_O\rfloor)_S\ \ (proposition)_T$
$\Rightarrow\quad \texttt{for } O\ \ \texttt{for } S\lceil O \to \mathbf{n}(O)\rfloor\ \ T$

[1] таким чином, $\mathbf{N}$ є певним мета-нетерміналом

Продовжуючи попередній приклад, ці правила призводять до наступних перетворень (для полегшення читання ми додаємо дужки):

`every member M of some committee C declares some view O such that every member N of C supports O`

$\Downarrow$ (2a)

`for (every member M of some committee C) M declares some view O such that every member N of C supports O`

$\Downarrow$ (2a)

`for (every member M of some committee C) M declares some view O such that for (every member N of C) N supports O`

$\Downarrow$ (2b)

`for (some committee C) for (every member M of C) M declares some view O such that for(every member N of C) N supports O`

Після застосування правил (1a-1c, 2a-2b) до пропозиції S, ми отримуємо пропозицію в *пласкій нормальній формі*, позначену $S^{\dagger}$. Ми використовуємо пласкі нормальні форми, щоб визначити перетворення цільової пропозиції в розділі (1.3.4).

3. Нормалізація атрибутів. Тепер перепишемо атрибути понять в єдину уніфіковану форму. Наступні правила застосовуються до входжень нетерміналів *notion*, *nounNotion* та *possessed*.

   (a) $(\{\, leftAttribute \,\})_L\ (\mathbf{N})_O\ (isPredicate\ \{\, \texttt{and}\ isPredicate \,\})_A$
   $\Rightarrow\quad L\ O\ \texttt{such that}\ \mathbf{n}(O)\ \texttt{is}\ A$

   (b) $(\{\, leftAttribute \,\})_L\ (\mathbf{N})_O$
   $\texttt{that}\ (doesPredicate\ \{\, \texttt{and}\ doesPredicate \,\})_V$
   $\Rightarrow\quad L\ O\ \texttt{such that}\ \mathbf{n}(O)\ V$

   (c) $(\{\, leftAttribute \,\})_L\ (leftAttribute)_A\ (\mathbf{N})_O$
   $[\, \texttt{such that}\ (proposition)_S \,]$
   $\Rightarrow\quad L\ O\ \texttt{such that}\ \mathbf{n}(O)\ \texttt{is}\ A\ [\, \texttt{and}\ S \,]$

За допомогою цих правил, всі предикати переводяться під прості пропозиції. Розглянемо, наприклад, наступні перетворення:

`prime natural number X that divides N`
$\Downarrow$ (3b)
`prime natural number X such that X divides N`
$\Downarrow$ (3c)
`prime number X such that X is natural and X divides N`
$\Downarrow$ (3c)
`number X such that X is prime and X is natural and X divides N`

4. Розщеплення складних предикатів. Тепер позбудемось кон'юнкцій в простих твердженнях. Наступні правила застосовуються до входжень *simpleProposition*:

   (a) $(terms)_O$ $(doesPredicate)_P$ `and` $(doesPredicate$ { `and` $doesPredicate$ $\})_T$
   $\Rightarrow$ O P `and` O T

   (b) $(terms)_O$ `is` $(isPredicate)_P$ `and` $(isPredicate$ { `and` $isPredicate$ $\})_T$
   $\Rightarrow$ O `is` P `and` O `is` T

   (c) $(terms)_O$ `is` $(is\_aPredicate)_P$ `and` $(is\_aPredicate$ { `and` $is\_aPredicate$ $\})_T$
   $\Rightarrow$ O `is` P `and` O `is` T

   Відмітимо, що в цих правилах, всі терми в $(terms)_O$ є простими.

5. Елімінація пропозицій "there exists". В подальшому, $\bar{O}$ позначає входження *notion*, *primClassNoun* або *primPossessedNoun* з видаленим списком імен $\mathbf{n}(O)$. В цій групі правил, ми переводимо "there is"-пропозиції в квантифіковані пропозиції з необмеженими кванторами. Наступні правила застосовуються до входжень *thereIsProposition*:

(a) there exists $(notionSymbol)_O$
$[\,\texttt{such that}\ (proposition)_S\,]\ [\,,\ (notions)_T\,]$
$\Rightarrow$ for some $\mathbf{n}(O)$ $(O\ [\,\texttt{and}\ S\,]\ [\,\texttt{and there exists}\ T\,])$

(b) there exists no $(notionSymbol)_O\ [\,\texttt{such that}\ (proposition)_S\,]$
$\Rightarrow$ for every $\mathbf{n}(O)$ (not $(O\ [\,\texttt{and}\ S\,])$)

(c) there exists $(notion)_O\ [\,,\ (notions)_T\,]$
$\Rightarrow$ for some $\mathbf{n}(O)$ ($\mathbf{n}(O)$ is $\bar{O}\ [\,\texttt{and there exists}\ T\,]$)

(d) there exists no $(notion)_O$
$\Rightarrow$ for every $\mathbf{n}(O)$ ($\mathbf{n}(O)$ is not $\bar{O}$)

Проілюструємо ці правила наступними перетвореннями:

there exists number E and prime divisors G,H of E
$\Downarrow$ (1c,3c)
there exists number E, divisors G,H of E
such that G,H is prime
$\Downarrow$ (5c)
for some E (E is a number and there exists
divisors G,H of E such that G,H is prime)
$\Downarrow$ (5c)
for some E (E is a number and for some G,H
(G,H is divisors of E such that G,H is prime))

6. Перетворення квантіфікованих понять II. Наступна група правил видаляє квантифіковані поняття з решти входжень нетерміналу *term*. Подібно до правил (2a-2b), ці правила застосовуються до входжень *simpleProposition* and *headProposition*. В антецеденті правила O є першим зліва нативним входженням $quantifiedNotion$ в S. В консеквенті, O замінюється в S на $\mathbf{n}(O)$:

(a) $(terms)_T\ (doesPredicate\lceil(quantifiedNotion)_O\rfloor)_S$
$\Rightarrow$ for $O\ \ T\ \ S\lceil O \rightarrow \mathbf{n}(O)\rfloor$

(b) `for` $(quantifiedNotion\lceil(quantifiedNotion)_O\rfloor)_S$ $(proposition)_T$
$\Rightarrow$ `for` O `for` $S\lceil O \rightarrow \mathbf{n}(O)\rfloor$ T

Попередній приклад продовжується наступним чином:

`for (some committee C) for (every member M of C) M declares some view O such that for(every member N of C) N supports O`

$\Downarrow$ (6a)

`for (some committee C) for (every member M of C) for (some view O such that for (every member N of C) N supports O) M declares O`

7. Обробка кванторів. Попередня група правил перетворила всі юніти-терми на прості терми. Тепер ми перетворимо обмежені квантори ForTheL на необмежені квантори над імплікаціями та кон'юнкціями. Наступні правила застосовуються до входжень *headProposition* (7a-7f) та *symbProposition* (7g-7h):

(a) `for [(] every` $(notionSymbol)_O$ `[such that` $(proposition)_T$ `] [)]` $(proposition)_S$
$\Rightarrow$ `for every` $\mathbf{n}(O)$ `(if` O `[and` T `] then` S`)`

(b) `for [(] some` $(notionSymbol)_O$ `[such that` $(proposition)_T$ `] [)]` $(proposition)_S$
$\Rightarrow$ `for some` $\mathbf{n}(O)$ `(`O `[and` T `] and` S`)`

(c) `for [(] no` $(notionSymbol)_O$ `[such that` $(proposition)_T$ `] [)]` $(proposition)_S$
$\Rightarrow$ `for every` $\mathbf{n}(O)$ `(if` O `[and` T `] then (not` S`))`

(d) `for [(] every` $(notion)_O$ `[)]` $(proposition)_S$
$\Rightarrow$ `for every` $\mathbf{n}(O)$ `(if` $\mathbf{n}(O)$ `is` $\bar{O}$ `then` S`)`

(e) `for [(] some` $(notion)_O$ `[)]` $(proposition)_S$
$\Rightarrow$ `for some` $\mathbf{n}(O)$ `(`$\mathbf{n}(O)$ `is` $\bar{O}$ `and` S`)`

(f) `for [(] no` $(notion)_O$ `[)]` $(proposition)_S$
$\Rightarrow$ `for every` $\mathbf{n}(O)$ `(if` $\mathbf{n}(O)$ `is` $\bar{O}$ `then (not` S`))`

(g) `forall` $(notionSymbol)_O$ $(symbProposition)_S$
$\Rightarrow$ `forall` $\mathbf{n}(O)$ `(`O `=>` S`)`

(h) `exists` $(notionSymbol)_O$ $(symbProposition)_S$
$\Rightarrow$ `exists` $\mathbf{n}(O)$ `(`O `/\` S`)`

8. Елімінація "has"-предикатів. Розглянемо входження O нетерміналу $primPossessedNoun$ та входження N нетерміналу $term$. Позначимо через $O\lceil N\rfloor$ початковий загальний або визначений іменник, де N поставлено на відновлене перше аргументне місце. Наприклад, якщо O є `element` та N є `S`, тоді $O\lceil N\rfloor$ є `element of S`.

Наступні правила застосовуються до входжень $simpleProposition$:

(a) $(terms)_N$ `is (with|of|having)` $(hasPredicate)_S$
$\Rightarrow$ N `has` S

(b) $(term)_N$ `,` $(term$ `{ ,` $term$ `}`$)_T$ `has` $(hasPredicate)_S$
$\Rightarrow$ N `has` S `and` T `has` S

(c) $(term)_N$ `has` $(primPossessedNoun)_O$`[such that` $(proposition)_S$`]`
`[and (`$possessed$ `{ and` $possessed$ `})`$_T$ `]`
$\Rightarrow$ `for some` $\mathbf{n}(O)$ `(`$\mathbf{n}(O)$ `is` $\bar{O}\lceil N\rfloor$ `[and` S`][and` N `has` T`])`

(d) $(term)_N$ `has no` $(primPossessedNoun)_O$
`[such that` $(proposition)_S$ `]`
$\Rightarrow$ `for every` $\mathbf{n}(O)$ `(not (`$\mathbf{n}(O)$ `is` $\bar{O}\lceil N\rfloor$ `[and` S `]))`

Продемонструємо ці правила на наступному прикладі:

```
A has a wife W and a salary not enough for W
```
$\Downarrow$ (1a,1c,3a)
```
A has wife W and salary S such that S is not enough for W
```
$\Downarrow$ (8c)
```
for some W (W is wife of A and
A has salary S such that S is not enough for W)
```
$\Downarrow$ (8c)
```
for some W (W is wife of A and
```

`for some S (S is salary of A and S is not enough for W))`

Тут, `wife of _` є примітивом загального іменника, що утворює поняття, а `salary of _` є примітивом визначеного іменника, що утворює функцію. Таким чином, в останній пропозиції маємо входження нетерміналу $is\_aPredicate$ обох видів.

9. Обробка "is a"-предикатів. Розглянемо входження $nounNotion$ вигляду $((primClassNoun)_O$ `such that` $(proposition)_S)$. Згідно з граматикою, $\mathbf{n}(O)$ містить щонайбільше одне ім'я, і ця властивість зберігається впродовж попередніх перетворень. Нехай N буде входженням нетерміналу $terms$. Якщо $\mathbf{n}(O)$ пусте, то вираз $S\lceil\mathbf{n}(O) \to N\rfloor$ дорівнює S. Якщо $\mathbf{n}(O)$ містить ім'я v, то $S\lceil\mathbf{n}(O) \to N\rfloor$ є результатом підстановки N замість кожного входження v в S.

   Наступні правила застосовуються до входжень $simpleProposition$:

   (a) $(term)_N$ `,` $(term$ `{ , }` $term)_T$ `is` $([$ `not` $])_C$ $(nounNotion)_O$
   $\Rightarrow$ N `is` C O `and` T `is` C O
   (якщо O не має в атрібуті мультипредиката, застосованого до $\mathbf{n}(O)$)

   (b) $(terms)_N$ `is` $([$ `not` $])_C$ $(primClassNoun)_O$
   $[$ `such that` $(proposition)_S$ $]$
   $\Rightarrow$ `(`C `(`N `is` $\bar{O}$ $[$ `and` $S\lceil\mathbf{n}(O) \to N\rfloor$ $]$ `))`

   (c) $(term)_N$ `,` $(term$ `{ , }` $term)_T$ `is` $(primClassNoun)_O$
   $\Rightarrow$ N `is` O `and` T `is` O

   (d) $(terms)_N$ `is` $([$ `not` $])_C$ $(definiteTerm)_O$
   $\Rightarrow$ N `is` C `equal to` O

   Щоб пояснити умову в першому правилі, розглянемо два ланцюга переписувать. В першому прикладі, "is a"-предикат не містить мультипримітивів в атрібуті. Тому він застосовується незалежно до кожного підмета:

   `A,B are not natural numbers`
   $\Downarrow$ (1a,1c,3c)

`A,B is not number X such that X is natural`

$\Downarrow$ (9a)

`A is not number X such that X is natural`
`and B is not number X such that X is natural`

$\Downarrow$ (9b)

`(not (A is number and A is natural))`
`and (not (B is number and B is natural))`

В другому прикладі, "is a"-предикат використовує мультипримітиви:

`A,B are not equal numbers`

$\Downarrow$ (1a,1c,3c)

`A,B is not number X such that X is equal`

$\Downarrow$ (9b)

`not (A,B is number and A,B is equal)`

$\Downarrow$ (9c)

`not (A is number and B is number and A,B is equal)`

Відмітимо також, що входження N в правилі (9b) містить більше одного терма лише тоді, коли S містить мультипредикати, застосовані до $\mathbf{n}(O)$. В цьому випадку, початкове поняття взагалі не має імен, тобто ім'я в $\mathbf{n}(O)$ було введено правилом (1a). Тому, $\mathbf{n}(O)$ входить в S лише як підмет простої пропозиції (завдяки правилам (3a-3c)), а отже вираз $S\lceil\mathbf{n}(O) \to N\rfloor$ є правильно побудованим.

10. Елімінація мультипредикатів. Розглянемо входження S деякого мультиприміттиву та входження N нетерміналу *term*. Позначимо через $S\lceil N\rfloor$ початкове дієслово або прикметник, де N поставлено на відновлене перше аргументне місце. Наприклад, якщо S є `parallel` та N є `X`, то $S\lceil N\rfloor$ є `parallel to X`.

Наступні правила застосовуються до входжень *simpleProposition*:

(a) $(term)_N$ [`does`] [`not`] [`pairwise`] $primVerbM$
$\Rightarrow$ syntax error

(b) $(term)_N$ `is [not] [pairwise]` $primAdjectiveM$
$\Rightarrow$ syntax error

(c) $(terms)_N$ `[does] not ([pairwise])`$_P$ $(primVerbM)_S$
$\Rightarrow$ `not (`$N\ P\ S$`)`

(d) $(terms)_N$ `is not ([pairwise])`$_P$ $(primAdjectiveM)_S$
$\Rightarrow$ `not (`N `is` $P\ S$`)`

(e) $(term)_{N_1}$ `,` ... `,` $(term)_{N_n}$ `[does] pairwise` $(primVerbM)_S$
$\Rightarrow$ `(`$N_1\ S\lceil N_2\rfloor$`) and ... (`$N_i\ S\lceil N_{i+j}\rfloor$`) ... and (`$N_{n-1}\ S\lceil N_n\rfloor$`)`

(f) $(term)_{N_1}$ `,` ... `,` $(term)_{N_n}$ `[does]` $(primVerbM)_S$
$\Rightarrow$ `(`$N_1\ S\lceil N_2\rfloor$`) and ... (`$N_i\ S\lceil N_{i+1}\rfloor$`) ... and (`$N_{n-1}\ S\lceil N_n\rfloor$`)`

(g) $(term)_{N_1}$ `,` ... `,` $(term)_{N_n}$ `is pairwise` $(primAdjectiveM)_S$
$\Rightarrow$ `(`N_1 `is` $S\lceil N_2\rfloor$`) and ... (`N_i `is` $S\lceil N_{i+j}\rfloor$`) ...`
`and (`N_{n-1} `is` $S\lceil N_n\rfloor$`)`

(h) $(term)_{N_1}$ `,` ... `,` $(term)_{N_n}$ `is` $(primAdjectiveM)_S$
$\Rightarrow$ `(`N_1 `is` $S\lceil N_2\rfloor$`) and ... (`N_i `is` $S\lceil N_{i+1}\rfloor$`) ...`
`and (`N_{n-1} `is` $S\lceil N_n\rfloor$`)`

11. Обробка звичайних та символьних предикатів. Тепер перетворимо атомарні пропозиції з входженнями *terms* та *symbTerms* на кон'юнкції. Наступні правила застосовуються до входжень *simpleProposition* (11a–11b) та *primPredicateSymbol* (11c):

(a) $(term)_N$ `[ ,` $(terms)_T$ `] [does] ([not])`$_C$ $(primVerb)_S$
$\Rightarrow$ `(`C `(`$N\ S$`)) [and`$\ T\ C\ S$`]`

(b) $(term)_N$ `[ ,` $(terms)_T$ `] is ([not])`$_C$ $(primAdjective)_S$
$\Rightarrow$ `(`C `(`N `is` S`)) [and`$\ T$ `is`$\ C\ S$`]`

(c) $(primPredicateSymbol\lceil(symbTerm)_N$ `,` $(symbTerms)_T\rfloor)_S$
$\Rightarrow$ $S\lceil N\rfloor$ `/\` $S\lceil T\rfloor$

12. Обробка синонімів. Юніт, що ми отримали після застосування попередніх правил, є фактично формулою першого порядку, з атомами

п'яти можливих форм (ми розглядаємо n-арний загальний іменник, прикметник та дієслово як предикатний символ арності $(n+1)$):

term [`does`] *primVerb*	*term* `is` *primAdjective*
term `is` *primClassNoun*	*primPredicateSymbol*
specialProposition	

Всі терми в цій формулі складаються з визначених іменників, функціональних символів та змінних, і отже є простими. Наступне правило застосовується до тих примітивів, які були введені як символи. Ми підставляємо замість кожного такого примітива відповідне інстанціювання позначуваного юніта і заново застосовуємо всю процедуру переписування до отриманої пропозиції. Ця рекурсія не може потрапити до нескінченого циклу, оскільки інтродуктори лінійно впорядковані в ForTheL-тексті.

Наприклад, інтродуктори

```
[a relation on D @ a relation with the domain equal to D]
[U ** V @ the intersection of U with V]
```

призводять до наступних перетворень:

```
X is relation on A ** B ** C
```
$\Downarrow$ (12)
```
X is a relation with the domain equal to
the intersection of A with the intersection of B with C
```
$\Downarrow$ (1–12)
```
X is relation and domain of X is equal to
intersection of A with intersection of B with C
```

Тепер перетворення завершено, і ми маємо правильно побудовану формулу першого порядку, що є формульним образом початкової пропозиції. Приклад повного ланцюга перетворень наведено на рисунку 2.

for all nonequal points A,B there exists a straight line L such that A and B lie on L and any straight line that contains A and contains B is L

⇓ (1a,1c)

for every nonequal points A,B there exists straight line L such that A,B lies on L and every straight line M that contains A and contains B is L

⇓ (2a)

for every nonequal points A,B there exists straight line L such that A,B lies on L and for every straight line M that contains A and contains B M is L

⇓ (3b)

for every nonequal points A,B there exists straight line L such that A,B lies on L and for every straight line M such that M contains A and contains B M is L

⇓ (3c)

for every points A,B such that A,B is nonequal there exists line L such that L is straight and A,B lies on L and for every line M such that M is straight and M contains A and contains B M is L

⇓ (4a)

for every points A,B such that A,B is nonequal there exists line L such that L is straight and A,B lies on L and for every line M such that M is straight and M contains A and M contains B M is L

⇓ (5c)

for every points A,B such that A,B is nonequal for some L (L is line such that L is straight and A,B lies on L and for every line M such that M is straight and M contains A and M contains B M is L)

⇓ (7d,7e)

for every A,B (if A,B is points such that A,B is nonequal then for some L (L is line such that L is straight and A,B lies on L and for every M (if M is line such that M is straight and M contains A and M contains B then M is L)))

⇓ (9b,9c,9d)

for every A,B (if A is points and B is points and A,B is nonequal then for some L (L is line and L is straight and A,B lies on L and for every M (if M is line and M is straight and M contains A and M contains B then M is equal to L)))

⇓ (10h)

for every A,B (if A is points and B is points and A is nonequal to B then for some L (L is line and L is straight and A,B lies on L and for every M (if M is line and M is straight and M contains A and M contains B then M is equal to L)))

⇓ (11a)

for every A,B (if A is points and B is points and A is nonequal to B then for some L (L is line and L is straight and A lies on L and B lies on L and for every M (if M is line and M is straight and M contains A and M contains B then M is equal to L)))

⇓ (12, маючи [x contains/contain y @ y lies on x])

for every A,B (if A is points and B is points and A is nonequal to B then for some L (L is line and L is straight and A lies on L and B lies on L and for every M (if M is line and M is straight and A lies on M and B lies on M then M is equal to L)))

Рис. 2: Отримання формульного образа пропозиції

1.2.7. Декларація змінних

Вільні змінні в пропозиції S — це змінні, що є вільними в її формульному образі: $\mathcal{FV}(S) = \mathcal{FV}(|S|)$. Так само, зв'язані змінні в пропозиції S — це змінні, що зв'язані в $|S|$: $\mathcal{BV}(S) = \mathcal{BV}(|S|)$. Правила ForTheL забороняють пропозиції, чиї формульні образи містять вкладені квантори по одній змінній. Наприклад, пропозиція `every number n divides some number n` не є правильно побудованою.

Неформально кажучи, пропозиція S *описує* змінну $v \in \mathcal{FV}(S)$, якщо з S випливає належність v до класу, визначеного деяким поняттям. Наступні визначення дають формальну експлікацію цьому терміну:

Терм t називається *позитивним*, якщо t не є квантифікованим поняттям вигляду `no` *notion* і аргументи t є позитивними термами.

Синтаксичний примітив I називається *дескриптивним*, якщо має місце одна з наступних умов:

- I є прикметником `equal to` arg_1;
- I є прикметником, дієсловом або предикатним символом; I введено як синонім; зразок в інтродукторі починається з деякої змінної v, а позначувана пропозиція описує v.

Предикат P називається *дескриптивним*, якщо P не є запереченням і має місце одна з наступних умов:

- P є "is a"-предикатом, утвореним з поняття з позитивними аргументами;
- P є "is a"-предикатом, утвореним з позитивного визначеного терму;
- P побудовано на дескриптивному примітиві прикметника або дієслова з позитивними аргументами;
- P скомпоновано з декількох атомарних предикатів, один з яких є дескриптивним.

Нарешті, пропозиція S *описує* змінну v, якщо виконується одна з наступних умов:

- S побудована з дескриптивного предикатного символа, де в першому аргументі стоїть послідовність змінних, що містить v;
- S є простою пропозицією з дескриптивним предикатом, де в підметі стоїть послідовність змінних, що містить v;
- S є кон'юнкцією декількох пропозицій, одна з яких описує v.

Наприклад, маючи наступні інтродуктори

```
[x belongs/belong to y @ x is an element of y]
[x « y @ x belongs to y]
```

пропозиція `u,v belong to some finite (s << W) and Q is a subset of no set` описує змінні `u`, `v`, але не змінну `Q`. Зауважимо, що предикатний символ `<<` породжує символ поняття, оскільки позначувана пропозиція `x belongs to y` описує змінну `x`.

1.2.8. Пропозиції визначення та сигнатури

Пропозиції ForTheL, що використовуються в розділах визначення та в сигнатурних розділах, мають спеціальну форму. Синтаксис пропозицій визначення описується наступними продукціями:

$$defProposition \rightarrow notionDef \mid functionDef \mid predicateDef$$

$$functionDef \rightarrow functionSym \texttt{ is equal to } plainTerm$$

$$\begin{aligned} functionSym \rightarrow\ & [\texttt{ the }]\ primDefiniteNoun \\ \mid\ & primInfixFunctionSymbol \\ \mid\ & primPrefixFunctionSymbol \\ \mid\ & primPostfixFunctionSymbol \end{aligned}$$

$$predicateDef \rightarrow predicateSym \texttt{ iff } proposition$$

$$
\begin{aligned}
\mathit{predicateSym} \;\rightarrow\; & \mathit{variable}\ \texttt{is}\ \mathit{primAdjective} \\
\mid\; & \mathit{variable}\ \texttt{,}\ \mathit{variable}\ \texttt{are}\ \mathit{primAdjectiveM} \\
\mid\; & \mathit{variable}\ \mathit{primVerb} \\
\mid\; & \mathit{variable}\ \texttt{,}\ \mathit{variable}\ \mathit{primVerbM} \\
\mid\; & \mathit{primPredicateSymbol}
\end{aligned}
$$

$$
\begin{aligned}
\mathit{notionDef} \;\rightarrow\; & [\,\texttt{a} \mid \texttt{an}\,]\ \mathit{primClassNoun}\ \texttt{is}\ [\,\texttt{a} \mid \texttt{an}\,]\ \mathit{notion} \\
\mid\; & \mathit{primNotionSymbol}\ \texttt{iff}\ \mathit{variable}\ \texttt{is}\ [\,\texttt{a} \mid \texttt{an}\,]\ \mathit{notion}
\end{aligned}
$$

Кожна пропозиція визначення S має так звану *головну зв'язку*, яка може бути рівністю (як в $functionDef$), еквівалентністю (як в другому типі $notionDef$ та в $predicateDef$) або словом `is` (як в першому типі $notionDef$). Юніт зліва від головної зв'язки називається *головою* визначення, юніт справа від неї — *тілом* визначення.

Пропозиція визначення S є правильно побудованою, якщо виконуються наступні умови:

- в голові визначення всі терми в аргументних місцях є попарно різними змінними;
- кожна змінна, що входить вільно в тіло визначення, входить також в голову;
- якщо примітив в голові визначення введено як синонім, то відповідний позначуваний юніт є примітивом з тими самими змінними в аргументних місцях;
- примітив в голові визначення не повинен входити в тіло визначення (рекурсивні визначення не підтримуються в сьогодняшній версії ForTheL); це стосується також позначуваного примітива, якщо голова визначення є синонімом;
- якщо S визначає поняття, воно повинно мати щонайбільше одне ім'я;
- якщо S визначає символьне поняття, ім'я, що вказано в голові, має бути підметом в тілі визначення.

Формульний образ пропозиції визначення формується так:

$$|\,(primClassNoun)_H\ \texttt{is}\ (notion)_B\,| = \forall v\,|\,v\ \texttt{is}\ \bar{H}\ \texttt{iff}\ v\ \texttt{is}\ B\,|$$
$$|\,(primNotionSymbol)_H\ \texttt{iff}\ ((variable)_v\ \texttt{is}\ notion)_B\,| = \forall v\,|\,H\ \texttt{iff}\ B\,|$$
$$|\,(functionDef)_S\,| = |\,(predicateDef)_S\,| = |S|$$

В першій рівності, змінна v є або $\mathbf{n}(H)$, або новою змінною, якщо $\mathbf{n}(H)$ пусте. В другій рівності, v є $\mathbf{n}(H)$ за умовою правильно побудованності. Рівності в останньому рядку означають, що визначення функцій та предикатов обробляються як звичайні пропозиції ForTheL.

Синтаксис пропозиції сигнатури визначається так:

$$sigProposition \rightarrow notionSig \mid functionSig \mid predicateSig$$

$$notionSig \rightarrow \texttt{notion}\ [\,\texttt{a}\,|\,\texttt{an}\,]\ primClassNoun$$

$$functionSig \rightarrow \texttt{function}\ functionSym$$

$$predicateSig \rightarrow \texttt{predicate}\ predicateSym$$

Як і в пропозиціях визначень, в пропозиції сигнатури юніт, що слідує за "видовим" словом, називається *головою*. Пропозиція сигнатури є правильно побудованою, якщо, по-перше, всі терми в аргументних місцях головного юніту є попарно різними змінними, і, по-друге, в разі, коли примітив в голові введено як синонім, то відповідний позначуваний юніт є примітивом з тими самими змінними в аргументних місцях.

Формульним образом пропозиції сигнатури є логічна істина:

$$|\,sigProposition\,| = \texttt{truth}$$

1.3. Розділи ForTheL

Почнемо з прикладу правильно побудованного ForTheL-тексту (номери рядков наведені для зручності і не належать до тексту).

```
1:    [a set/sets] [an element/elements of x]
```

```
 2:    [x is in y @ x is an element of y]
 3:    [x belongs/belong to y @ x is in y]
 4:    [a subset/subsets of x] [x is empty]
 5:    Definition DefSubset. Let S be a set.
 6:      A subset of S is a set X such that
 7:        every element of X belongs to S.
 8:    Definition DefEmpty. Let S be a set.
 9:      S is empty iff S has no elements.
10:    Axiom ExEmpty. There exists an empty set.
11:    Theorem. Let S be a set.
12:      S is a subset of every set iff S is empty.
13:    Proof.
14:      First let S be a subset of every set.
15:        We can show that S is empty.
16:          Let z belong to S.
17:          Take an empty set E.
18:          z is an element of E.
19:          We have a contradiction.
20:        end.
21:      end.
22:      Now assume that S is empty. Let T be a set.
23:        Every element of S is in T (by DefEmpty).
24:        Hence S is a subset of T (by DefSubset).
25:      end.
26:    qed.
```

1.3.1. Розділи верхнього рівня та речення

ForTheL-текст є послідовністю інтродукторів та *розділів верхнього рівня*. Розділами верхнього рівня є *аксіоми*, *визначення*, *теореми* та спеціальні *сигнатурні розділи*. Наведений вище текст містить чотири розділа верхнього рівня: визначення поняття підмножини (рядки 5–7), визна-

чення пустої множини (рядки 8–9), аксіому існування пустої множини (рядок 10), та теорему з доведенням (рядки 11–26).

Кожний розділ верхнього рівня має *заголовок*, що визначає тип розділу і може мати *мітку* для використання в *посиланнях* (рядки 23, 24).

$$
\begin{aligned}
text &\rightarrow \{\ axiom \mid definition \mid theorem \\
&\qquad\qquad \mid signature \mid introductor\ \} \\
axiom &\rightarrow axmHeader\ \{\, assume \,\}\ axmAffirm \\
axmHeader &\rightarrow \texttt{Axiom}\ [\, label \,]\ . \\
definition &\rightarrow defHeader\ \{\, assume \,\}\ defAffirm \\
defHeader &\rightarrow \texttt{Definition}\ [\, label \,]\ . \\
theorem &\rightarrow thmHeader\ \{\, assume \,\}\ thmAffirm \\
thmHeader &\rightarrow (\ \texttt{Theorem} \mid \texttt{Lemma} \mid \texttt{Corollary}\ )\ [\, label \,]\ . \\
signature &\rightarrow sigHeader\ \{\, assume \,\}\ sigAffirm \\
sigHeader &\rightarrow \texttt{Signature}\ [\, label \,]\ . \\
label &\rightarrow word
\end{aligned}
$$

Всередині розділів верхнього рівня ми зустрічаємо *речення*: *припущення* (рядки 5,8,11,14,16,22), *селекції* (line 17), решта є *твердженнями*. Кожний розділ верхнього рівня складається з нуля або більше припущень, за якими слідує твердження відповідного вигляду.

Припущення та твердження утворюються на основі пропозицій. В подальшому, будемо казати, що твердження (припущення) *стверджує* (відповідно, *припускає*) відповідну пропозицію. Селекції будуються з понять (*обирають* їх) та стверджують непустоту означених ними класів.

$$assume \rightarrow asmPrefix\ proposition\ .$$

$$
\begin{aligned}
asmPrefix &\rightarrow \texttt{let} \mid [\,\texttt{let us} \mid \texttt{we can}\,]\ (\,\texttt{assume} \mid \texttt{suppose}\,)\ [\,\texttt{that}\,] \\
select &\rightarrow selPrefix\ notions\ [\,ref\,]\ . \\
selPrefix &\rightarrow [\,\texttt{then} \mid \texttt{hence}\,]\ [\,\texttt{let us} \mid \texttt{we can}\,]\ (\,\texttt{take} \mid \texttt{choose}\,) \\
axmAffirm &\rightarrow proposition\ . \\
defAffirm &\rightarrow defProposition\ . \\
sigAffirm &\rightarrow [\,\texttt{let us} \mid \texttt{we can}\,]\ sigProposition\ . \\
thmAffirm &\rightarrow affPrefix\ proposition\ [\,ref\,]\ .\ [\,prfHeader\ proof\,] \\
&\quad \mid\ prfPrefix\ proposition\ [\,ref\,]\ .\ proof \\
affPrefix &\rightarrow [\,\texttt{then} \mid \texttt{hence}\,] \\
prfHeader &\rightarrow \texttt{proof}\ [\,\texttt{by}\ method\,]\ . \mid \texttt{indeed} \\
prfPrefix &\rightarrow [\,\texttt{let us} \mid \texttt{we can}\,]\ (\,\texttt{prove} \mid \texttt{show}\,)\ [\,\texttt{by}\ method\,]\ [\,\texttt{that}\,] \\
method &\rightarrow \texttt{contradiction} \mid \texttt{case analysis} \\
&\quad \mid\ \texttt{induction}\ [\,\texttt{on}\ plainTerm\,] \\
ref &\rightarrow \texttt{(}\ \texttt{by}\ label\ \{\ \texttt{,}\ label\,\}\ \texttt{)}
\end{aligned}
$$

1.3.2. Розділи доведення

В коректному тексті, кожне твердження (окрім тих, що входять в аксіоми та визначення) має бути обґрунтоване: воно має логічно випливати зі своїх логічних попередників в тексті. Автор тексту може зробити це випливання більш очевидним, або за допомогою посилань на релевантні розділи верхнього рівня, або супроводжуючи це твердження *розділом доведення*, який стане неявним логічним попередником для твердження. Приклад, наведений вище, містить два розділи доведення: один для твердження в рядку 12 (рядки 13–26), і один для твердження в рядку 15

(рядки 16–20).

Розділи доведення в ForTheL складаються з припущень, селекцій, тверджень (що можуть супроводжуватись власними доведеннями) та розділів нижнього рівня, таких як випадки або прості блоки. Доведення теореми в прикладі складається з двох простих блоків (рядки 14–21 та 22–25).

Прості блоки використовуються для структурізації доведення: вони визначають область дії припущень та декларації змінних. Прості блоки можуть зустрічатися будь-де в доведенні. Розбір випадків має завершувати розділ доведення. Розділи-випадки слідують один за одним; кожний розділ-випадок починається з пропозиції, яка є гіпотезою випадку.

$$\begin{aligned}
proof &\rightarrow [\,\{\,prfBody\,\}\;prfLast\,]\;qed \\
qed &\rightarrow \texttt{end.} \mid \texttt{qed.} \mid \texttt{obvious.} \mid \texttt{trivial.} \\
prfBody &\rightarrow assume \mid select \mid thmAffirm \mid block \\
prfLast &\rightarrow thmAffirm \mid block \mid case\,\{\,case\,\} \\
block &\rightarrow blkHeader\;proof \\
blkHeader &\rightarrow \texttt{Block}\;[\,label\,]\;\texttt{.} \mid \texttt{now} \mid \texttt{first} \mid \texttt{second} \mid \ldots \\
case &\rightarrow \texttt{Case}\;proposition\;[\,ref\,]\;\texttt{.}\;proof
\end{aligned}$$

Автор тексту може явно вказати метод доведення, який він застосовує: наразі, в системі підтримуються доведення від супротивного, розбором випадків та доведення по індукції. Примітки “by contradiction” або “by case analysis” можна опускати, оскільки система може встановити, що застосовано один з цих методів, за формою розділа доведення. Напроти, примітка “by induction” є необхідною, оскільки вона вказує верифікатору, що необхідно сформулювати *гипотезу індукції* і додати її в розділ доведення в необхідному місці.

Формалізація доведень по індукції в ForTheL заснована на *загальнo-*

му принципі індукції:

$$\forall\vec{x}(\mathrm{IH}(\vec{x}) \supset F[\vec{x}]) \supset \forall\vec{x}F[\vec{x}]$$

де $\mathrm{IH}(\vec{x}) = \forall\vec{y}((t[\vec{y}] \prec t[\vec{x}]) \supset F[\vec{y}])$, t є деяким термом першого порядку, а $\prec$ є фундованим впорядкуванням. Цей принцип може бути висловлено наступним чином: *при доведенні того, що твердження P має місце для деяких довільних але фіксованих значень $\vec{x}$, ми можемо безпечно припустити, що це твердження має місце для всіх значень $\vec{y}$, що менші за $\vec{x}$ відносно деякого фундованого впорядкування. Індукційний терм t* використовується для того, щоб "зібрати" набір значень в одне значення.

Отже, доведення по індукції інтерпретуються в припущенні відповідної гіпотези індукції. Ця гіпотеза формулюється автоматично і стає додатковим логічним попередником в доведенні.

Доведення по індукції повинно мати цільову пропозицію (див. 1.3.4) вигляду `for every` $(notion)_{O_1}$ ... `for every` $(notion)_{O_n}$ *proposition*. Якщо індукційний терм не вказано явно, за нього приймається перша змінна в $\mathbf{n}(O_1)$. Всі змінні, що входять в індукційний терм мають належати до $\mathbf{n}(O_1) \cup \cdots \cup \mathbf{n}(O_n)$ або бути декларованими ще до початку доведення. Гіпотеза індукції утворюється з цільової пропозиції наступним чином:

`for every` $(notion)_{O_1}$ ... `for every` $(notion)_{O_n}$ $(proposition)_S$
$\Rightarrow$ `for every` $O_1\sigma$ ... `for every` $O_n\sigma$ `if` $t\sigma$ `-<-` t `then` $S\sigma$

де t є індукційним термом, а σ є підстановкою, що перейменовує змінні в t, підставляючи замість них деякі нові змінні. Гіпотеза індукції неявно вставляється в доведення під час нормалізації (див. 1.3.4).

В логіці першого порядку ми не можемо дати скінчену аксіоматизацію фундованості впорядкування. Тому в ForTheL введено спеціальний бінарний предикатний символ `-<-`, який позначає абстрактне фундоване відношення в доведеннях по індукції. Автор тексту має самостійно сформулювати аксіоми, що визначать конкретні властивості цього відношення. Наприклад, для того, щоб проводити доведення по індукції по

натуральним числам, достатньо наступних аксіом (за умови, що традиційні аксіоми Пеано також надані в тексті):

```
Axiom ZeroOrSucc. For every natural number N  N = 0
   or there exists a natural number M such that N = succ M.

Axiom IndOrder. For every natural number N  N -<- succ N.
```

Справді, розглянемо просту лему, яку доведемо по індукції по `n`:

```
Lemma. For all natural numbers m,n
       (succ m) + n = succ (m + n).
```

Гіпотезою індукції (`IH`) для цієї леми є така пропозиція:

```
          for all natural numbers x,y if y -<- n
             then (succ x) + y = succ (x + y)
```

Тепер запишемо розділ доведення:

```
Proof by induction on n.
  Let m,n be natural numbers.
  ## починається область дії IH
  Case n = 0. Obvious.                        ## аксіоми Пеано
  Case n != 0.
    Take a natural number x
                    such that succ x = n. ## ZeroOrSucc
    (succ m) + n = succ ((succ m) + x).   ## аксіоми Пеано
    (succ m) + x = succ (m + x).          ## IH та IndOrder
    succ (m + x) = m + n.                 ## аксіоми Пеано
    Hence (succ m) + n = succ (m + n).    ## з попереднього
  end.
qed.
```

Аксіома `ZeroOrSucc` потрібна, щоб розбити доведення на базу і крок індукції. Аксіома `IndOrder` потрібна, що довести, що `x -<- n`, а отже, гіпотеза індукції може бути застосована. Відмітимо, що властивості впорядкування для відношення `-<-` не використовуються в доведенні.

1.3.3. Розділи-попередники та декларація змінних

Ми визначаємо межі розділів в тексті $\mathbb{T}$ згідно з правилами граматики, наведеними вище. Наприклад, будь-який розділ верхнього рівня починається з першою літерою свого заголовка і завершується там, де завершується твердження розділу. Також, твердження, що супроводжується доведенням, завершується там, де завершується доведення, а не крапкою в кінці речення.

Будемо казати, що розділ A *накриває* розділ B (B *міститься* в A), якщо B лежить між межами A і B не є самим розділом A. Так, твердження, що супроводжується розділом доведення накриває цей розділ і всі розділи всередині доведення. Назвемо A *елементом* B, якщо B є мінімальною секцією, що накриває A. Наприклад, будь-який простий блок або випадок має рівно один елемент, а саме, розділ доведення, прикріплений до цього блоку або випадку. Скажемо, що A є *синтаксичним попередником* B, якщо B починається розділу A.

Розділ A є *логічним попередником* розділу B, якщо A є максимальним (відносно накриття) синтаксичним попередником B. Будемо позначати множину логічних попередників розділу A в тексті $\mathbb{T}$ через $\mathbf{LP}_{\mathbb{T}}(A)$.

В прикладі на початку глави, твердження в рядку 15 має наступних логічних попередників: обидва визначення, аксіома `ExEmpty` та твердження в рядках 11 та 14. Твердження в рядку 12 не є логічним попередником даного твердження.

Припущення та селекції ForTheL можуть вводити в текст нові змінні. Ми визначаємо множину декларованих змінних згідно з формою припущення або селекції. Змінна v декларована у селекції, якщо серед перелічених в ній понять є поняття O, таке що $v \in \mathbf{n}(O)$. Позначивши множину декларованих змінних через $\mathbf{Decl}_{\mathbb{T}}(\cdot)$, маємо:

$$\mathbf{Decl}_{\mathbb{T}}(selPrefix\ \ (notions)_N\ \ [\,ref\,]\ .) = \mathbf{Decl}_{\mathbb{T}}(N)$$

$$\mathbf{Decl}_{\mathbb{T}}([\,\texttt{a}\mid\texttt{an}\,]\ \ (notion)_O\ \ [\,(\ ,\mid\texttt{and}\,)\ \ (notions)_N\,]) = \mathbf{n}(O)\ \ [\,\cup\,\mathbf{Decl}_{\mathbb{T}}(N)\,]$$

Припущення декларує ті змінні, що описані у припущеній пропозиції:

$$\mathbf{Decl}_{\mathbb{T}}(asmPrefix\ \ (proposition)_S\ .) = \{v \mid S \text{ описує } v\}$$

Для розділів, які не є припущеннями або селекціями, $\mathbf{Decl}_{\mathbb{T}}(A) = \varnothing$.

Змінні, що декларовані в реченні A, можуть вільно входити в A та в логічних послідовників A. Припущення може декларувати змінні, що вже були декларовані; наступні декларації не відміняють, а лише уточнюють попередні. Навпаки, змінні, що декларовані в селекції, не повинні бути декларовані в жодному логічному попереднику цієї селекції. Будемо позначати множину змінних, що декларовані в логічних попередниках розділу A через $\overline{\mathbf{Decl}}_{\mathbb{T}}(A)$. Такі змінні будемо називати *відомими* в A.

1.3.4. Нормалізація доведень

Ми запроваджуємо набір переписуючих правил, що нормалізують розділи доведення в ForTheL-тексті: елімінують спеціальні пропозиції `thesis` та `contrary`, оповнюють доведення, перекладають розділи-випадки в комбінації розділів доведення та простих блоків, додають гіпотези індукції. Застосовані до ForTheL-тексту $\mathbb{T}$, ці правила переводять його в *нормалізований* текст $\mathbb{T}^{\circ}$.

Обробка цільової пропозиції. Кожна теорема P в $\mathbb{T}$ замінюється теоремою $\langle P \rangle$, що визначається наступними рівностями:

$$\big\langle\, (thmHeader)_H\ (\{assume\})_Q\ (thmAffirm)_A \,\big\rangle \;=\; H\, Q\, \langle A \rangle_{\texttt{contradiction}}$$

Ми пропускаємо заголовок та припущення і нормалізуємо твердження теореми. Речення та розділи нижнього рівня нормалізуються стосовно *цільової пропозиції* (вказаної в нижньому індексі). Грубо кажучи, цільовою пропозицією для даного розділу є пропозиція, яка доводилась на початку цього розділу. Початкова цільова пропозиція — це `contradiction`,

тобто, на цей момент в тексті не доводиться жодна пропозиція.

$$\begin{aligned}
&\left\langle\, affPrefix\ (proposition)_S\ ([\, ref\,])_R\ .\ [\,(prfHeader)_H\ (proof)_P\,]\,\right\rangle_T\ = \\
&\qquad\qquad = \ S\lceil T\rfloor\ R\ .\ [\,H\ \langle P\rangle^{\top}_{S\lceil T\rfloor^{\dagger}}\,] \\
&\left\langle\, (prfPrefix)_H\ (proposition)_S\ ([\, ref\,])_R\ .\ (proof)_P\,\right\rangle_T\ = \\
&\qquad\qquad = \ H\ S\lceil T\rfloor\ R\ .\ \langle P\rangle^{\top}_{S\lceil T\rfloor^{\dagger}}
\end{aligned}$$

Тут і надалі, $S\lceil T\rfloor$ позначає юніт S, де входження спеціальних пропозицій `thesis` та `contrary` замінені на пропозицію T та, відповідно, на її заперечення. Пласка нормальна форма пропозиції, що доводиться, стає цільовою пропозицією для супроводжуючого розділу доведення. Також, супроводжуюче доведення вважається *мотивованим* розділом (позначається знаком $\top$ у верхньому індексі).

$$\begin{aligned}
&\left\langle\, asmPrefix\ (proposition)_S\ .\,\right\rangle_T\ = \ \texttt{assume}\ S\lceil T\rfloor\ . \\
&\left\langle\, selPrefix\ (notions)_S\ ([\, ref\,])_R\ .\,\right\rangle_T\ = \ \texttt{take}\ S\lceil T\rfloor\ R\ . \\
&\left\langle\, (blkHeader)_H\ (proof)_P\,\right\rangle_T\ = \ H\ \langle P\rangle^{\bot}_T \\
&\left\langle\, \texttt{Case}\ (proposition)_S\ ([\, ref\,])_R\ .\ (proof)_P\,\right\rangle_T\ = \\
&\qquad = \ \left\langle\, \texttt{(if}\ S\lceil\texttt{contradiction}\rfloor\ \texttt{then thesis)}\ R\ . \right. \\
&\qquad\qquad\qquad \left.\texttt{proof. assume}\ S\lceil\texttt{contradiction}\rfloor\ .\ P\,\right\rangle_T
\end{aligned}$$

Цільова пропозиція простого блоку просто передається всередину блоку. Прості блоки вважаються немотивованими розділами (знак $\bot$ в верхньому індексі). Розділ-випадок переписується в твердження поточної цільової пропозиції; це твердження супроводжується доведенням. Спеціальні пропозиції `thesis` та `contrary` не повинні зустрічатися в гіпотезі випадку.

$$\begin{aligned}
&\left\langle\, (assume)_A\ (proof)_P\,\right\rangle^{\top}_T\ = \ \left\langle\, \texttt{(Block. }A\ P\texttt{) end.}\,\right\rangle^{\top}_T \qquad (T\backslash A = T) \\
&\left\langle\, (assume)_A\ (proof)_P\,\right\rangle^{s}_T\ = \ \langle A\rangle_T\ \langle P\rangle^{s}_{T\backslash A} \qquad \text{(інакше)}
\end{aligned}$$

В розділі доведення, кожне припущення певним чином співвідноситься з поточною цільовою пропозицією та породжує нову “скорочену” цільову

пропозицію для подальшого доведення. Вище ми позначили цю редукцію через $T\backslash A$. Якщо в мотивованому розділі зустрічається припущення, яке не скорочує поточну ціль, тобто $T\backslash A = T$, ми вважаємо, що решта розділу є немотивованою, і заключаємо її в простий блок.

Для поточної цільової пропозиції T та припущення A з пропозицією S, ми визначаємо нову ціль $T\backslash A$ наступним чином. Зауважимо одразу, що якщо T знаходиться в пласкій нормальній формі, то $T\backslash A$ також.

- якщо T має вигляд $(\texttt{if}\ S\lceil T\rfloor\ \texttt{then}\ R)$ то $T\backslash A = R$;
- якщо $S\lceil T\rfloor$ має вигляд $(\texttt{not}\ T)$ то $T\backslash A = \texttt{contradiction}$. Зокрема, це трапляється, коли A є припущенням `assume the contrary.`;
- якщо $T = (\texttt{for all}\ (notion)_{O_1} \ldots \texttt{for all}\ (notion)_{O_n}(proposition)_R)$, аргументи понять в $O_1, \ldots, O_n$ є простими термами (зокрема, це має місце, коли T знаходиться в пласкій нормальній формі), $S\lceil T\rfloor$ має вигляд $(\mathbf{n}(O_i)\ \texttt{is}\ \bar{O}_i)$, $\mathbf{n}(O_i)$ не перетинається з $\overline{\mathbf{Decl}}_{\mathbb{T}}(A)$ — тоді $T\backslash A$ є пропозицією

  $$\texttt{for all}\ O_1\ \ldots\ \texttt{for all}\ O_{i-1}\ \texttt{for all}\ O_{i+1}\ \ldots\ \texttt{for all}\ O_n\ R$$

- в решті випадків, $T\backslash A = T$.

Селекції, твердження та прості блоки не впливають на ціль:

$$\begin{aligned}
&\left\langle\ (select \mid thmAffirm \mid block)_A\ (proof)_P\ \right\rangle^s_T\ =\ \left\langle A\right\rangle_T\ \left\langle P\right\rangle^s_T\\
&\left\langle\ (case)_{C_1}\ \ldots\ (case)_{C_n}\ qed\ \right\rangle^s_T\ =\ (S_i \text{ позначає гіпотезу випадка в } C_i)\\
&\qquad\qquad =\ T\ .\ \texttt{proof.}\ \left\langle C_1\right\rangle_T\ \ \ldots\ \ \left\langle C_n\right\rangle_T\\
&\qquad\qquad\qquad \left\langle S_1\ \texttt{or}\ \ldots\ \texttt{or}\ S_n.\right\rangle_{\texttt{contradiction}}\ \ \texttt{end. end.}
\end{aligned}$$

Послідовність розділів-випадків, що завершують розділ доведення трансформується в твердження поточної цілі, супроводжене доведенням. Нагадаємо, що спеціальні пропозиції `thesis` та `contrary` не повинні зустрічатися в гіпотезі випадку.

$$\left\langle\ qed\ \right\rangle^{\top}_T\ =\ T\ .\ \texttt{end.} \qquad\qquad \left\langle\ qed\ \right\rangle^{\bot}_T\ =\ \texttt{end.}$$

Наприкінці мотивованого доведення достатньо довести поточну ціль для того, щоб довести початкове твердження. Тому ми додаємо цільову пропозицію в розділ і завершуємо його. В простому блоці (в немотивованому доведенні) такого твердження немає, отже ми просто залишаємо розділ.

Перенесення посилань. В тексті $\mathbb{T}'$, який отримано з попередніх перетворень, кожний розділ доведення, що супроводжує твердження (а не простий блок), також завершується твердженням. За побудовою, це завершальне твердження не має посилання. Для кожного твердження A, яке має посилання і розділ доведення P, ми переносимо посилання з A до твердження B, яке завершує P.

Додання гіпотези індукції. В тексті $\mathbb{T}''$, який отримано з попередніх перетворень, ми розглядаємо кожне твердження A, супроводжене доведенням P по індукції по простому терму t (не обов'язково вказаному явно). Як вже було сказано (1.3.2), A повинно стверджувати пропозицію, чия пласка нормальна форма має вигляд

$$\texttt{for every } (notion)_{O_1} \ \ldots \ \texttt{for every } (notion)_{O_n} \ (proposition)_S$$

і змінні з t повинні належати до $(\mathbf{n}(O_1) \cup \cdots \cup \mathbf{n}(O_n)) \cup \overline{\mathbf{Decl}}_{\mathbb{T}''}(A)$.

Нехай I буде відповідною гіпотезою індукції. Всі змінні з t, що декларовані всередині P, мають бути декларовані припущеннями, а не селекціями. Якщо серед розділів-елементів P знайдеться такий, що всі змінні з t в ньому відомі, тоді речення `assume that` I`.` вставляється в P перед першим таким елементом.

На цьому перетворення завершується і ми отримуємо нормалізований текст $\mathbb{T}^\circ$. Розглянемо як приклад лему з підрозділу 1.3.2. В нормалізованому вигляді ця лема виглядатиме так:

```
Lemma. For all natural numbers m,n (succ m) + n = succ (m + n).
   Proof by induction on n.
     Let m,n be natural numbers.
     Assume that for all natural numbers x,y
```

```
      if y -<- n then (succ x) + y = succ (x + y).
    (succ m) + n = succ (m + n).
    proof.
      if n = 0 then (succ m) + n = succ (m + n).
      proof.
          assume n = 0.
          (succ m) + n = succ (m + n).
      end.
      if n != 0 then (succ m) + n = succ (m + n).
      proof.
          assume n != 0.
          Take a natural number x such that succ x = n.
          (succ m) + n = succ ((succ m) + x).
          (succ m) + x = succ (m + x).
          succ (m + x) = m + n.
          Hence (succ m) + n = succ (m + n).
          (succ m) + n = succ (m + n).
      end.
      n = 0 or n != 0.
    end.
  end.
```

Якщо ми випустимо базу індукції з початкового доведення леми:

```
Lemma. For all natural numbers m,n (succ m) + n = succ (m + n).
  Proof by induction on n.
    Let m,n be natural numbers.
    Assume that n != 0.
    Take a natural number x such that succ x = n.
    (succ m) + n = succ ((succ m) + x).
    (succ m) + x = succ (m + x).
    succ (m + x) = m + n.
    Hence (succ m) + n = succ (m + n).
  qed.
```

то нормалізований текст виглядатиме наступним чином:

```
Lemma. For all natural numbers m,n (succ m) + n = succ (m + n).
   Proof by induction on n.
     Let m,n be natural numbers.
     Assume that for all natural numbers x,y
        if y -<- n then (succ x) + y = succ (x + y).
     Block.
        Assume that n != 0.
        Take a natural number x such that succ x = n.
        (succ m) + n = succ ((succ m) + x).
        (succ m) + x = succ (m + x).
        succ (m + x) = m + n.
        Hence (succ m) + n = succ (m + n).
     end.
     (succ m) + n = succ (m + n).
   end.
```

1.3.5. Формульний образ розділів

Подібно до пропозицій ForTheL, будь-який розділ P в нормалізованому тексті $\mathbb{T}^\circ$ може бути перекладений в формулу першого порядку $|P|$ — *формульний образ* P.

1. Припущення: $|\,(asmPrefix\ (proposition)_S\ .)_P\,| = |S|$

2. Селекції: $|\,selPrefix\ (notions)_N\ .\,| = F'$

   Формула F' утворюється наступним чином. Нехай F буде формульним образом пропозиції `there exist` N. Для кожного поняття O в послідовності N такого, що $\mathbf{n}(O)$ непусте, ми видаляємо відповідний квантор існування по $\mathbf{n}(O)$ з F, і таким чином звільнюємо змінні з $\mathbf{n}(O)$. Отриману формулу ми й позначаємо F'. Розглянемо наступні

приклади:

$$| \texttt{Take a set S and an element of S.} | = \\ = (\mathrm{aSet}(S) \wedge \exists x\, (\mathrm{aElement}(x, S)))$$

$$| \texttt{Take a set S and an element x of S.} | = \\ = (\mathrm{aSet}(S) \wedge \mathrm{aElement}(x, S))$$

$$| \texttt{Take a set and an element x of S.} | = \\ = \exists z\, (\mathrm{aSet}(z) \wedge \mathrm{aElement}(x, S))$$

3. Твердження:

$$| (proposition)_S \ . | = |S|$$
$$| (defProposition)_S \ . | = | (sigProposition)_S \ . | = |S|$$
$$| affPrefix \ (proposition)_S \ [\, ref \,] \ . \ [\, prfHeader \ proof \,] | = |S|$$
$$| prfPrefix \ (proposition)_S \ [\, ref \,] \ . \ proof | = |S|$$

4. Доведення ($\vec{v}_A = \mathbf{Decl}_{\mathbb{T}^\circ}(A) \backslash \overline{\mathbf{Decl}}_{\mathbb{T}^\circ}(A)$):

$$| (assume)_A \ (proof)_P | = \forall \vec{v}_A\, (|A| \supset |P|)$$
$$| (select)_A \ (proof)_P | = \exists \vec{v}_A\, (|A| \wedge |P|)$$
$$| (prfAffirm)_A \ (proof)_P | = |A| \wedge |P|$$
$$| (block)_A \ (proof)_P | = |A| \wedge |P|$$
$$| qed | = verum$$

5. Прості блоки: $| blkHeader \ \ (proof)_P | = |P|$

6. Розділи верхнього рівня ($\vec{v}_A = \mathbf{Decl}_{\mathbb{T}^\circ}(A) \backslash \overline{\mathbf{Decl}}_{\mathbb{T}^\circ}(A)$):

$$| axmHeader \ (assume)_{A_1} \ \ldots \ (assume)_{A_n} \ (axmAffirm)_B | = \\ = | sigHeader \ (assume)_{A_1} \ \ldots \ (assume)_{A_n} \ (sigAffirm)_B | = \\ = | defHeader \ (assume)_{A_1} \ \ldots \ (assume)_{A_n} \ (defAffirm)_B | = \\ = | thmHeader \ (assume)_{A_1} \ \ldots \ (assume)_{A_n} \ (thmAffirm)_B | = \\ = \forall \vec{v}_{A_1}\, (|A_1| \supset \ldots \supset \forall \vec{v}_{A_n} (|A_n| \supset |B|) \ldots)$$

Звернемо увагу на те, що розділ доведення, що супроводжує твердження, не впливає на формульний образ цього твердження. Формульний образ виражає логічний зміст розділу, і розділ доведення поруч з твердженням не обов'язково відповідає змісту твердження.

Зауважимо також, що формульний образ сигнатурного розділу є просто логічною істиною. Сигнатурні розділи не мають власного логічного змісту, вони лише окреслюють області визначеності примітивів, для яких не наведено визначень. Сигнатурні розділи використовуються для забезпечення онтологічної коректності тексту.

Як і раніше, ми визначаємо множину вільних (зв'язаних) змінних розділу A як множину вільних (зв'язаних) змінних формульного образу A: $\mathcal{FV}(A) = \mathcal{FV}(|A|)$ and $\mathcal{BV}(A) = \mathcal{BV}(|A|)$. Правильно побудований розділ A в нормалізованому ForTheL-тексті має задовольняти наступним умовам:

- $\mathcal{FV}(A) \subseteq \mathbf{Decl}_{\mathbb{T}^\circ}(A) \cup \overline{\mathbf{Decl}}_{\mathbb{T}^\circ}(A)$
- $\mathcal{BV}(A) \cap (\mathbf{Decl}_{\mathbb{T}^\circ}(A) \cup \overline{\mathbf{Decl}}_{\mathbb{T}^\circ}(A)) = \varnothing$
- якщо A є селекцією, то $\mathbf{Decl}_{\mathbb{T}^\circ}(A) \cap \overline{\mathbf{Decl}}_{\mathbb{T}^\circ}(A) = \varnothing$
- якщо A є твердженням в визначенні або в сигнатурному розділі, то всі змінні з $\overline{\mathbf{Decl}}_{\mathbb{T}^\circ}(A)$ повинні входити в головний юніт A.

РОЗДІЛ 2
ПРОВЕДЕННЯ МАТЕМАТИЧНИХ МІРКУВАНЬ

2.1. Загальні означення

Ми розглядаємо логіку першого порядку з рівністю $\approx$, логічними зв'язками заперечення $\neg$, кон'юнкції $\wedge$, диз'юнкції $\vee$, імплікації $\supset$, еквівалентності $\equiv$, істини $\top$, хибності $\bot$, кванторами загальності $\forall$ та існування $\exists$.

Підстановки ми розуміємо як функції, що співставляють змінним терми. Для будь-якої підстановки ϕ, якщо $x\phi \neq x$, терм $x\phi$ називається *субститутом* в ϕ. Підстановка *скінчена* тоді і тільки тоді, коли множина її субститутів скінчена. Скінчені підстановки ми записуємо як послідовності виду $[t_1/x_1, \ldots, t_n/x_n]$.

Позиція є словом в алфавіті $\{0, 1, \ldots\}$. У подальшому грецькі літери τ, μ та ν позначають позиції; ε позначає нульову позицію (пусте слово). *Множина позицій* у термі t, позначена $\Pi(t)$, визначається наступним чином (нижче, $i.\Pi$ позначає $\{\, i.\tau \mid \tau \in \Pi \,\}$):

$$\Pi(f(s_0, \ldots, s_n)) = \{\varepsilon\} \cup 0.\Pi(s_0) \cup \ldots \cup n.\Pi(s_n) \qquad \Pi(c) = \{\varepsilon\}$$

Множина позицій у формулі F, позначена $\Pi(F)$, є об'єднанням *множини позитивних позицій* $\Pi^+(F)$ та *множини негативних позицій* $\Pi^-(F)$:

$$\begin{aligned}
\Pi(F) &= \Pi^+(F) \cup \Pi^-(F) \\
\Pi^+(F \equiv G) &= \{\varepsilon\} \cup 0.\Pi(F) \cup 1.\Pi(G) \\
\Pi^-(F \equiv G) &= 0.\Pi(F) \cup 1.\Pi(G) \\
\Pi^+(F \supset G) &= \{\varepsilon\} \cup 0.\Pi^-(F) \cup 1.\Pi^+(G) \\
\Pi^-(F \supset G) &= 0.\Pi^+(F) \cup 1.\Pi^-(G) \\
\Pi^+(F \wedge G) &= \{\varepsilon\} \cup 0.\Pi^+(F) \cup 1.\Pi^+(G) \\
\Pi^-(F \wedge G) &= 0.\Pi^-(F) \cup 1.\Pi^-(G) \\
\Pi^+(F \vee G) &= \{\varepsilon\} \cup 0.\Pi^+(F) \cup 1.\Pi^+(G) \\
\Pi^-(F \vee G) &= 0.\Pi^-(F) \cup 1.\Pi^-(G)
\end{aligned}$$

$$\Pi^+(\forall x\, F) = \{\varepsilon\} \cup 0.\Pi^+(F) \qquad \Pi^-(\forall x\, F) = 0.\Pi^-(F)$$
$$\Pi^+(\exists x\, F) = \{\varepsilon\} \cup 0.\Pi^+(F) \qquad \Pi^-(\exists x\, F) = 0.\Pi^-(F)$$
$$\Pi^+(\neg F) = \{\varepsilon\} \cup 0.\Pi^-(F) \qquad \Pi^-(\neg F) = 0.\Pi^+(F)$$
$$\Pi^+(\top) = \{\varepsilon\} \qquad \Pi^-(\top) = \varnothing$$
$$\Pi^+(\bot) = \{\varepsilon\} \qquad \Pi^-(\bot) = \varnothing$$
$$\Pi^+(P(s_0, \ldots, s_n)) = \{\varepsilon\} \cup \bigcup i.\Pi(s_i) \qquad \Pi^-(P(s_0, \ldots, s_n)) = \varnothing$$

Зауважимо, що множини позитивних та негативних позицій можуть перетинатись: входження під зв'язкою еквівалентності вважаються одночасно позитивними та негативними.

Будемо казати, що π *передує* τ ($\pi \lll \tau$), коли $\pi = \omega.i.\mu$ та $\tau = \omega.j.\eta$ для деяких ω, μ, η та $i < j$. Такі позиції будемо називати *суміжними*.

Маючи терм t та позицію $\tau \in \Pi(t)$, ми позначаємо через $t|_\tau$ підтерм терма t в позиції τ: $t|_\varepsilon = t$, $f(s_0, \ldots, s_n)|_{i.\tau} = s_i|_\tau$. Маючи формулу F та позицію $\tau \in \Pi(F)$, ми позначаємо через $F|_\tau$ підформулу або підтерм формули F в позиції τ: $F|_\varepsilon = F$, $(*\, F)|_{0.\tau} = F|_\tau$, $(F * G)|_{0.\tau} = F|_\tau$, $(F * G)|_{1.\tau} = G|_\tau$, $P(s_0, \ldots, s_n)|_{i.\tau} = s_i|_\tau$. Тут $(*\, F)$ позначає $(\neg F)$, $(\forall x\, F)$ або $(\exists x\, F)$; $(F\ *\ G)$ позначає $(F \equiv G)$, $(F \supset G)$, $(F \wedge G)$, або $(F \vee G)$.

Маючи терм t, позицію $\tau \in \Pi(t)$ та терм p, терм $t[p]_\tau$ є результатом заміни підтерму $t|_\tau$ на p:

$$t[p]_\varepsilon = p \qquad f(s_0, \ldots, s_n)[p]_{i.\tau} = f(s_0, \ldots, s_i[p]_\tau, \ldots, s_n)$$

Маючи формулу F, позицію підформули (чи підтерму) $\tau \in \Pi(F)$ та формулу (чи терм) H, формула $F[H]_\tau$ є результатом заміни $F|_\tau$ на H за наступними правилами:

$$F[H]_\varepsilon = H \qquad (*\, F)[H]_{0.\tau} = (*\, F[H]_\tau)$$
$$(F * G)[H]_{0.\tau} = (F[H]_\tau * G) \qquad (F * G)[H]_{1.\tau} = (F * G[H]_\tau)$$
$$P(s_0, \ldots, s_n)[H]_{i.\tau} = P(s_0, \ldots, s_i[p]_\tau, \ldots, s_n)$$

Вільні змінні H можуть опинитись зв'язаними в $F[H]_\tau$. Інколи ми пишемо $F[H^+]_\tau$ ($F[H^-]_\tau$), вказуючи на те, що $\tau \in \Pi^+(F)$ ($\tau \in \Pi^-(F)$, відповідно).

2.2. Локально істинні твердження

У задачах автоматичного пошуку доведення, що походять з реальної математики, як правило, не зустрічаються абсолютно універсальні твердження, визначення та правила. Звичайно, все, що використовується, має певні умови чинності (такі як обмеження типу або обмеження області визначеності), які мають бути задовільнені. Наприклад, ми не можемо скоротити дріб $\frac{xy}{x}$, поки не доведено, що x не дорівнює нулю.

Візьмемо велику формулу виду $(\cdots \forall x\, (x \in \mathbb{R}^+ \supset (\cdots \frac{xy}{x} \cdots)) \cdots)$. Зрозуміло, що ми можемо замінити $\frac{xy}{x}$ на y, але як це довести? Саме питання здається абсурдним: оскільки терм містить зв'язані змінні, ми не можемо проводити жодних міркувань щодо нього. З традиційної точки зору, необхідно перш розщепити велику формулу до квантора, що зв'язує x, зробити підстановку (або сколемізувати), відділити $x \in \mathbb{R}^+$ і лише тоді робити скорочення.

Можна заперечити, що немає сенсу робити будь-які спрощення, коли спрощувана формула знаходиться десь всередині іншої формули. Рано чи пізно, але все одно велика формула має бути розщеплена, аби цей дріб міг бути використаний у доведенні. Тим не менш, ми вважаємо за корисне і повчальне спростити задачу *у її початковій формі* наскільки можливо, для того, щоб обрати найбільш перспективний напрямок пошуку доведення.

Окрім спрощення, є також проблема всюду визначеності даної формули. Кожне входження символу, введеного за визначенням, має задовольняти умовам визначення, щоб вважатися коректним. Бажано пересвідчитись, що задача сформульована коректно, до того, як братися за її вирішення. Але така верифікація знов-таки пов'язана з міркуваннями щодо термів, які містять зв'язані змінні (як у попередньому прикладі).

Та сама проблема існує і на пропозиційному рівні. Припустимо, що теорія, яка розглядається, містить аксіому $A \supset B$, де A та B — пропозиційні літери, і розглянемо формулу вигляду $(\cdots (A \wedge (\cdots B \cdots)) \cdots)$. Знов-таки, було б корисно замінити B на $\top$ (істина) і спростити формулу в цілому.

Повернемось до початкового прикладу. Справді, твердження "x не нуль" є безглуздим, але можна сказати "x не нуль у даному входженні дробу $\frac{xy}{x}$". Інтуіція підказує, що разом зі звичайним істиностним значенням існує так зване *локальне істиностне значення*, що визчається відносно до контексту, до деякої позиції у формулі. Твердження, що є, взагалі кажучі, хибним або навіть абсурдним, може стати локально істинним, якщо розглядати його у цій позиції.

Для довільних формули F, позиції $\pi \in \Pi(F)$ та формули U, ми визначаємо *локальний образ U у позиції π в формулі F* (позначається $\langle U \rangle_\pi^F$) за наступними тотожностями:

$$\begin{array}{lll}
\langle U \rangle_\varepsilon^F = U & \langle U \rangle_\varepsilon^\top = U & \langle U \rangle_\varepsilon^\bot = U \\
\langle U \rangle_\pi^{P(\vec{s})} = U & \langle U \rangle_{0.\pi}^{F \wedge G} = G \supset \langle U \rangle_\pi^F & \langle U \rangle_{1.\pi}^{F \wedge G} = F \supset \langle U \rangle_\pi^G \\
\langle U \rangle_{0.\pi}^{\neg F} = \langle U \rangle_\pi^F & \langle U \rangle_{0.\pi}^{F \vee G} = G \vee \langle U \rangle_\pi^F & \langle U \rangle_{1.\pi}^{F \vee G} = F \vee \langle U \rangle_\pi^G \\
\langle U \rangle_{0.\pi}^{\forall x F} = \forall\, x\, \langle U \rangle_\pi^F & \langle U \rangle_{0.\pi}^{F \supset G} = G \vee \langle U \rangle_\pi^F & \langle U \rangle_{1.\pi}^{F \supset G} = F \supset \langle U \rangle_\pi^G \\
\langle U \rangle_{0.\pi}^{\exists x F} = \forall\, x\, \langle U \rangle_\pi^F & \langle U \rangle_{0.\pi}^{F \equiv G} = \langle U \rangle_\pi^F & \langle U \rangle_{1.\pi}^{F \equiv G} = \langle U \rangle_\pi^G
\end{array}$$

Формула $\langle U \rangle_\pi^F$ виражає твердження "*U є істинною у позиції π у F*". Відмітимо, що ця формула не залежить від підформули (підтерму) $F|_\pi$.

Приклад 2.1. Нехай F є формулою

$$\forall\, x\, (x \in \mathbb{N} \supset \forall\, n\, (n \in \mathbb{N} \supset (x \approx \mathbf{fib}(n) \equiv \\ \equiv ((n \leq 1 \wedge x \approx 1) \vee x \approx (\mathbf{fib}(n-1) + \mathbf{fib}(n-2))))))$$

Ця формула представляє рекурсивне визначення — інакше кажучи, вона містить входження символу, що визначається, з правого боку від "головного" знаку еквівалентності. Справді, для $\pi = 0.1.0.1.1.1$, $F|_\pi = (x \approx (\mathbf{fib}(n-1) + \mathbf{fib}(n-2)))$. Треба переконатися, що аргументи $(n-1)$ та $(n-2)$ задовольняють умовам визначення та є строго меншими за n.

Зосередимось на другому аргументі $(n-2)$. Необхідно довести, що $\langle (n-2) \in \mathbb{N} \wedge (n-2) < n \rangle_\pi^F$. Ця формула, за визначенням, дорівнює

$$\forall\, x (x \in \mathbb{N} \supset \forall\, n (n \in \mathbb{N} \supset ((n \leq 1 \wedge x \approx 1) \vee ((n-2) \in \mathbb{N} \wedge (n-2) < n))))$$

Але ця формула є хибною за умови $n = x = 0$. Це вказує на помилку у запропонованому визначенні: $x = 0$ спростовує ліву частину диз'юнкції $F|_{0.1.0.1.1}$, отже ми маємо розглядати праву частину при $n = 0$ для того, щоб визначити істиностне значення усієї диз'юнкції. Тепер нескладно побудувати коректне визначення F':

$$\forall x\,(x \in \mathbb{N} \supset \forall n\,(n \in \mathbb{N} \supset (x \approx \mathbf{fib}(n) \equiv \\ \equiv ((n \leq 1 \wedge x \approx 1) \vee (n \geq 2 \wedge x \approx (\mathbf{fib}(n-1) + \mathbf{fib}(n-2)))))))$$

Розглянемо властивості локальних образів.

Лема 2.2. *Для довільних F, $\pi \in \Pi(F)$ та U, $\forall U \models \langle U \rangle_\pi^F$.*

Тут вираз $\forall U$ позначає універсальне замикання формули U. Щоб пересвідчитись в справедливості леми 2.2, достатньо побачити, що формула $\langle U \rangle_\pi^F$ еквівалентна певній диз'юнкції (під кванторами загальності), і формула U є позитивною компонентою цієї диз'юнкції.

Лема 2.3 (локальний *modus ponens*)**.** $\models \langle U \supset V \rangle_\pi^F \supset (\langle U \rangle_\pi^F \supset \langle V \rangle_\pi^F)$

Цю лему нескладно довести індукцією по структурі формули F, подібно до того, як ми доводимо нижче теорему 2.7.

Щоб проілюструвати вагомість цих двох результатів, наведемо три висновки з них:

Наслідок 2.4. $\models \langle U \equiv V \rangle_\pi^F \supset (\langle U \rangle_\pi^F \equiv \langle V \rangle_\pi^F)$

Доведення. За лемою 2.2 маємо $\models \langle (U \equiv V) \supset (U \supset V) \rangle_\pi^F$, а отже, за лемою 2.3, $\models \langle (U \equiv V) \rangle_\pi^F \supset (\langle U \supset V \rangle_\pi^F)$, і, знов за правилом локального *modus ponens*, $\models \langle (U \equiv V) \rangle_\pi^F \supset (\langle U \rangle_\pi^F \supset \langle V \rangle_\pi^F)$. Цілком аналогічно доводиться, що $\models \langle (U \equiv V) \rangle_\pi^F \supset (\langle V \rangle_\pi^F \supset \langle U \rangle_\pi^F)$. □

Наслідок 2.5. $\models \langle U \wedge V \rangle_\pi^F \equiv (\langle U \rangle_\pi^F \wedge \langle V \rangle_\pi^F)$

Доведення. Щоб довести імплікацію зліва направо, ми починаємо з пропозиційних тавтологій $(U \wedge V) \supset U$ та $(U \wedge V) \supset V$. Щоб довести імплікацію справа наліво, ми починаємо з пропозиційної тавтології $U \supset$

$(V \supset (U \wedge V))$. В кожному випадку, ми "переводимо" тавтологію всередину формули F за допомогою леми 2.2, а потім застосовуємо правило локального *modus ponens* за лемою 2.3. □

Наслідок 2.6. *Для довільного безкванторного контексту C,*

$$\models \big(\langle U_1 \equiv V_1\rangle_\pi^F \wedge \cdots \wedge \langle U_n \equiv V_n\rangle_\pi^F \wedge \langle t_1 \approx s_1\rangle_\pi^F \wedge \cdots \wedge \langle t_m \approx s_m\rangle_\pi^F\big) \supset \\ \supset \langle C[U_1, \ldots, U_n, t_1, \ldots, t_m] \equiv C[V_1, \ldots, V_n, s_1, \ldots, s_m]\rangle_\pi^F$$

Під контекстом тут розуміється формула "з дирками", куди можуть бути поміщені формули або терми, доповнюючи таким чином контекст до правильно побудованої формули першого порядку. Доведення аналогічне до попередніх тверджень.

Ці леми показують, що локальна істинність має цілком адекватні логічні властивості: будь-яка істина є локальною істиною, а правило *modus ponens* є локально застосовним. Але головна властивість локальних образів дається наступною теоремою.

Теорема 2.7. *Для довільної формули F, позиції підформули $\pi \in \Pi(F)$, та формул U і V:*

$$\models \langle U \equiv V\rangle_\pi^F \supset (F[U]_\pi \equiv F[V]_\pi)$$

Доведення. Доведемо теорему індукцією по довжині позиції π. Для бази індукії ($\pi = \varepsilon$), твердження теореми очевидне. Для кроку індукції розглянемо чотири основні випадки. В кожному випадку ми формально виводимо твердження теореми з гіпотези індукції та визначення локального образу.

Випадок $F = (\neg G)$, $\pi = 0.\tau$ (аналогічно для $F = (G \equiv H)$):

$$\models \langle U \equiv V\rangle_\tau^G \supset (G[U]_\tau \equiv G[V]_\tau) \quad \Rightarrow \\ \Rightarrow \quad \models \langle U \equiv V\rangle_\tau^G \supset (\neg G[U]_\tau \equiv \neg G[V]_\tau) \quad \Rightarrow \\ \Rightarrow \quad \models \langle U \equiv V\rangle_\pi^F \supset (F[U]_\pi \equiv F[V]_\pi)$$

Випадок $F = (G \supset H)$, $\pi = 0.\tau$ (аналогічно для $F = (G \vee H)$):

$$\begin{aligned}
&\models \langle U \equiv V\rangle_\tau^G \supset (G[U]_\tau \equiv G[V]_\tau) \qquad \Rightarrow \\
&\quad \Rightarrow \qquad \models (H \vee \langle U \equiv V\rangle_\tau^G) \supset (H \vee (G[U]_\tau \equiv G[V]_\tau)) \qquad \Rightarrow \\
&\quad \Rightarrow \qquad \models \langle U \equiv V\rangle_\pi^F \supset ((G[U]_\tau \supset H) \equiv (G[V]_\tau \supset H)) \qquad \Rightarrow \\
&\qquad\qquad\qquad\qquad \Rightarrow \qquad \models \langle U \equiv V\rangle_\pi^F \supset (F[U]_\pi \equiv F[V]_\pi)
\end{aligned}$$

Випадок $F = (H \supset G)$, $\pi = 1.\tau$ (аналогічно для $F = (G \wedge H)$):

$$\begin{aligned}
&\models \langle U \equiv V\rangle_\tau^G \supset (G[U]_\tau \equiv G[V]_\tau) \qquad \Rightarrow \\
&\quad \Rightarrow \qquad \models (H \supset \langle U \equiv V\rangle_\tau^G) \supset (H \supset (G[U]_\tau \equiv G[V]_\tau)) \qquad \Rightarrow \\
&\quad \Rightarrow \qquad \models \langle U \equiv V\rangle_\pi^F \supset ((H \supset G[U]_\tau) \equiv (H \supset G[V]_\tau)) \qquad \Rightarrow \\
&\qquad\qquad\qquad\qquad \Rightarrow \qquad \models \langle U \equiv V\rangle_\pi^F \supset (F[U]_\pi \equiv F[V]_\pi)
\end{aligned}$$

Випадок $F = (\exists\, x\, G)$, $\pi = 0.\tau$ (аналогічно для $F = (\forall\, x\, G)$):

$$\begin{aligned}
&\models \langle U \equiv V\rangle_\tau^G \supset (G[U]_\tau \equiv G[V]_\tau) \qquad \Rightarrow \\
&\quad \Rightarrow \qquad \models \forall\, x\, \langle U \equiv V\rangle_\tau^G \supset \forall\, x\, (G[U]_\tau \equiv G[V]_\tau) \qquad \Rightarrow \\
&\quad \Rightarrow \qquad \models \langle U \equiv V\rangle_\pi^F \supset (\exists\, x\, G[U]_\tau \equiv \exists\, x\, G[V]_\tau) \qquad \Rightarrow \\
&\qquad\qquad\qquad\qquad \Rightarrow \qquad \models \langle U \equiv V\rangle_\pi^F \supset (F[U]_\pi \equiv F[V]_\pi)
\end{aligned}$$

□

Наслідок 2.8. *Для довільної формули* F*, позиції терму* $\pi \in \Pi(F)$*, та термів* s *і* t*:*

$$\models \langle s \approx t\rangle_\pi^F \supset (F[s]_\pi \equiv F[t]_\pi)$$

Випливає з попередньої теореми та наслідку 2.6.

Наслідок 2.9.

$$\models \langle U\rangle_\pi^F \supset (F \equiv F[(U \wedge F|_\pi)]_\pi) \qquad \models \langle U\rangle_\pi^F \supset (F \equiv F[(U \supset F|_\pi)]_\pi)$$

Доведення. $\models U \supset (F|_\pi \equiv (U \wedge F|_\pi))$ та $\models U \supset (F|_\pi \equiv (U \supset F|_\pi))$. □

Отже ми можемо безпечно замінювати підформули не лише на еквівалентні їм формули але також на *локально еквівалентні*. Зауважимо, що твердження, зворотнє до теореми 2.7, має місце у пропозиційній логіці: $\models_0 \langle U \equiv V\rangle^F_\pi \equiv (F[U]_\pi \equiv F[V]_\pi)$. Тут локальна еквівалентність є *критерієм* підстановочної еквівалентності. У логіці першого порядку це не так, про що свідчить еквівалентність $(\exists\, x\, x \approx 0) \equiv (\exists\, x\, x \not\approx 0)$.

З'ясуємо, чи варті довіри міркування з прикладу 2.1. Умовно назвемо *визначенням* замкнену формулу вигляду $\forall\, \vec{x}\, (C(\vec{x}) \supset (P(\vec{x}) \equiv D(\vec{x})))$, де P є предикатним символом, $\vec{x}$ — послідовністю різних змінних, C — *формулою-умовою*, яка не містить P, а D — *формулою-розкриттям*. Розглянемо деяке входження (F, π), де $F|_\pi = P(\vec{s})$. Якщо можна довести $\langle C(\vec{s})\rangle^F_\pi$, то маємо $\langle P(\vec{s}) \equiv D(\vec{s})\rangle^F_\pi$ (звісно, за умови, що визначення є частиною теорії). Тоді, за теоремою 2.7, ми можемо розкрити визначення, замінивши $P(\vec{s})$ у (F, π) на $D(\vec{s})$.

Повертаючись до прикладу 2.1, можна гарантувати, що таке розкриття завжди можливе (завдяки істиності $\langle n - 1 \in \mathbb{N} \wedge n - 2 \in \mathbb{N}\rangle^F_\pi$) та завжди скінчене (завдяки істиності $\langle n - 1 < n \wedge n - 2 < n\rangle^F_\pi$).

Зауважимо, що отримані результати є чинними у інтуіціоністський логіці. Більш того, визначення локального образу легко поширити на мову (уні)-модальної логіки: $\langle U\rangle^{\Box F}_{0.\pi} = \Box\langle U\rangle^F_\pi$ та $\langle U\rangle^{\Diamond F}_{0.\pi} = \Box\langle U\rangle^F_\pi$. Тоді всі наші твердження можуть бути доведені у модальній логіці **K**, отже у будь-якій нормальній модальній логіці [44]. Це одна з переваг запропонованого поняття локальної істинності, яка відрізняє його від попередніх формалізмів такого роду, наприклад [45] або [46].

Разом з тим викладене поняття локального образу має один недолік: воно не є інваріантним по відношенню до локально еквівалентних перетворень в суміжних позиціях.

Приклад 2.10. Оскільки $\langle A\rangle^{A\wedge A}_0$, то $(A \wedge A)$ еквівалентне $(\top \wedge A)$ за теоремою 2.7. Але $\langle A\rangle^{A\wedge A}_1$ також справедливе, тоді як $\langle A\rangle^{\top\wedge A}_1$ — хибне.

Взагалі кажучи, ми можемо побудувати формулу F, чиї дві підформули U та V забезпечують певні локальні властивості друг для друга.

Використовуючи ці властивості, ми можемо замінити U локально еквівалентною формулою U', але таким чином втратимо локальні властивості в V.

Це не має великого значення, коли йдеться про разові перетворення, наприклад, спрощення формули. Але разом з тим, є такі локальні властивості, які бажано зберігати протягом всього сеансу роботи з формальним текстом. Це стосується, насамперед, виконуваності умов визначень — раз перевірені для заданого входження визначеного символу, вони повинні зберігатися, так щоб ми могли в будь-який момент розкрити це визначення без додаткових перевірок.

Тому ми дещо змінюємо визначення локального образу таким чином, щоб лише підформули з передуючих позицій потрапляли в контекст. Це обмеження є досить природним, оскільки потрібні нам пропозиції (декларації змінних, обмеження) звичайно записуються перед "важливими" твердженнями.

Спрямований локальний образ U у позиції π в формулі F (позначається $\langle\!|U|\!\rangle^F_\pi$) визначається наступним чином:

$$
\begin{array}{lll}
\langle\!|U|\!\rangle^F_\varepsilon = U & \langle\!|U|\!\rangle^\top_\varepsilon = U & \langle\!|U|\!\rangle^\bot_\varepsilon = U \\
\langle\!|U|\!\rangle^{P(\vec{s})}_\pi = U & \langle\!|U|\!\rangle^{F\wedge G}_{0.\pi} = \langle\!|U|\!\rangle^F_\pi & \langle\!|U|\!\rangle^{F\wedge G}_{1.\pi} = F \supset \langle\!|U|\!\rangle^G_\pi \\
\langle\!|U|\!\rangle^{\neg F}_{0.\pi} = \langle\!|U|\!\rangle^F_\pi & \langle\!|U|\!\rangle^{F\vee G}_{0.\pi} = \langle\!|U|\!\rangle^F_\pi & \langle\!|U|\!\rangle^{F\vee G}_{1.\pi} = F \vee \langle\!|U|\!\rangle^G_\pi \\
\langle\!|U|\!\rangle^{\forall x F}_{0.\pi} = \forall\, x\, \langle\!|U|\!\rangle^F_\pi & \langle\!|U|\!\rangle^{F\supset G}_{0.\pi} = \langle\!|U|\!\rangle^F_\pi & \langle\!|U|\!\rangle^{F\supset G}_{1.\pi} = F \supset \langle\!|U|\!\rangle^G_\pi \\
\langle\!|U|\!\rangle^{\exists x F}_{0.\pi} = \forall\, x\, \langle\!|U|\!\rangle^F_\pi & \langle\!|U|\!\rangle^{F\equiv G}_{0.\pi} = \langle\!|U|\!\rangle^F_\pi & \langle\!|U|\!\rangle^{F\equiv G}_{1.\pi} = \langle\!|U|\!\rangle^G_\pi
\end{array}
$$

Перш за все, зауважимо, що всі попередні твердження, доведені для "неспрямованих" локальних образів, справедливі також і для спрямованих. В певному сенсі, спрямований образ є просто скороченням звичайного локального образу, з викресленням деяких умов та альтернатив. Це демонструється наступною тривіальною лемою.

Лема 2.11. *Для довільних F, $\pi \in \Pi(F)$ та U, $\models \langle\!|U|\!\rangle^F_\pi \supset \langle U\rangle^F_\pi$.*

Доведемо тепер шукану "стійкість" спрямованих образів.

Теорема 2.12. *Для довільної формули F, позиції підформули $\pi \in \Pi(F)$ та суміжної з нею позиції $\tau \in \Pi(F)$,*

$$\models \langle\!|U \equiv V|\!\rangle_{\pi}^{F} \supset \left(\langle\!|W|\!\rangle_{\tau}^{F[U]_{\pi}} \equiv \langle\!|W|\!\rangle_{\tau}^{F[V]_{\pi}}\right)$$

Доведення. Легко бачити, що якщо τ передує π, то формули $\langle\!|W|\!\rangle_{\tau}^{F[U]_{\pi}}$ та $\langle\!|W|\!\rangle_{\tau}^{F[V]_{\pi}}$ ідентичні. Припустимо тепер, що $\pi \lll \tau$, тобто, існують такі ω, μ, та η, що $\pi = \omega.0.\mu$ та $\tau = \omega.1.\eta$. Повторюючи шлях доведення теореми 2.7 (її "спрямованої" версії), ми можемо звести задачу до

$$\models \langle\!|U \equiv V|\!\rangle_{0.\mu}^{G*H} \supset \left(\langle\!|W|\!\rangle_{1.\eta}^{(G*H)[U]_{0.\mu}} \equiv \langle\!|W|\!\rangle_{1.\eta}^{(G*H)[V]_{0.\mu}}\right)$$

де $(G * H) = F|_{\omega}$. Останнє твердження еквівалентне твердженню

$$\models \langle\!|U \equiv V|\!\rangle_{\mu}^{G} \supset \left(\langle\!|W|\!\rangle_{1.\eta}^{G[U]_{\mu}*H} \equiv \langle\!|W|\!\rangle_{1.\eta}^{G[V]_{\mu}*H}\right)$$

а звідти і твердженню

$$\models \langle\!|U \equiv V|\!\rangle_{\mu}^{G} \supset \left((G[U]_{\mu} \star \langle\!|W|\!\rangle_{\eta}^{H}) \equiv (G[V]_{\mu} \star \langle\!|W|\!\rangle_{\eta}^{H})\right)$$

де $\star$ є або $\supset$, або $\vee$, в залежності від $*$. За "спрямованою" теоремою 2.7, з $\langle\!|U \equiv V|\!\rangle_{\mu}^{G}$ випливає $(G[U]_{\mu} \equiv G[V]_{\mu})$, і теорему доведено. □

Наслідок 2.13. *Для довільної формули F, позиції терму $\pi \in \Pi(F)$ та суміжної з нею позиції $\tau \in \Pi(F)$,*

$$\models \langle\!|s \approx t|\!\rangle_{\pi}^{F} \supset \left(\langle\!|W|\!\rangle_{\tau}^{F[s]_{\pi}} \equiv \langle\!|W|\!\rangle_{\tau}^{F[t]_{\pi}}\right)$$

2.3. Відомості про входження термів

Ще один технічний прийом, що у стає у нагоді під час обробки тексту — це накопичення відомостей, пов'язаних з окремими входженнями термів. Ми кажемо, що літера L є "відомістю" про деяке входження терма t (йдеться про входження в формульний образ речення в нормалізованому ForTheL-тексті), якщо, по-перше, t є одним з аргументів L, по-друге, L є локально істинною в позиції цього входження, і, по-третє, ця локальна істиність може бути встановлена певним простим чином.

Спочатку формалізуємо цю "просту процедуру" встановлення локальної істиності твердження в наступному секвенціальному численні **S**:

$$\frac{}{L \to L} \qquad \frac{\Gamma \to G}{\Gamma, F \to G} \qquad \frac{\Gamma \to L[t/x]}{\Gamma, s \approx t \to L[s/x]}$$

$$\frac{\Gamma, F \to G}{\Gamma \to F \supset G} \qquad \frac{\Gamma, \neg F \to G}{\Gamma \to F \vee G} \qquad \frac{\Gamma \to G}{\Gamma \to \forall z\, G}$$

$$\frac{\Gamma, F[t/x] \to G}{\Gamma, \forall x\, F \to G} \qquad \frac{\Gamma, F, H \to G}{\Gamma, F \wedge H \to G} \qquad \frac{\Gamma, L \to G}{\Gamma, \neg\neg L \to G}$$

$$\frac{\Gamma, F, L_1, \ldots, L_n \to G}{\Gamma, (L_1 \wedge \cdots \wedge L_n) \supset F, L_1, \ldots, L_n \to G}$$

$$\frac{\Gamma, F, \neg L_1, \ldots, \neg L_n \to G}{\Gamma, (L_1 \vee \cdots \vee L_n) \vee F, \neg L_1, \ldots, \neg L_n \to G}$$

В наведених правилах виведення ми нехтуємо порядком формул в посилках. Символи L, L_1, $\ldots$, L_n позначають літери, символи F,G,H — формули, символ Γ — множину формул. Змінна z в правилі введення квантору загальності в цілі не повинна входити в посилки.

Числення **S** є, очевидно, коректним. Легко побачити, що проблема вивідності в цьому численні алгоритмічно розв'язна. Справді, кожен зворотній крок в процесі пошуку виведення від цільової секвенції до аксіом строго зменшує кількість логічних символів в секвенції. Правило введення квантору загальності в посилках не становить тут проблеми: потрібні підстановки отримуються за допомогою метазмінних та уніфікації.

Тепер опишемо процедуру породження відомостей.

Нехай маємо правильно побудований ForTheL-текст $\mathbb{T}$. Розглянемо деяке речення P в нормалізованому тексті $\mathbb{T}^\circ$. Нехай $\pi \in \Pi(|P|)$ є позицією терму t в формульному образі речення P. Будемо вважати, що всім аргументам t вже співставлені деякі літери-відомості, і позначимо їхню сукупну множину $\bar{\mathbb{L}}$ (якщо t є змінною або константою, $\bar{\mathbb{L}} = \varnothing$).

Літера L вважається *відомістю порядку* 0 про задане входження t, якщо t є одним з аргументів L, і секвенція $\bar{\mathbb{L}} \to \langle\!|L|\!\rangle_\pi^{|P|}$ може бути виведена в **S**. Позначимо множину усіх відомостей порядку 0 через $\mathbb{L}^0$.

Візьмемо найближчий до P логічний попередник, позначимо його Q. Літера L вважається *відомостю порядку* 1 про задане входження t, якщо t є одним з аргументів L, і секвенція $\bar{\mathbb{L}}, \mathbb{L}^0, |Q| \to \langle\!|L|\!\rangle_\pi^{|P|}$ може бути виведена в $\mathbf{S}$. Позначимо множину усіх відомостей порядку 1 через $\mathbb{L}^1$.

Візьмемо найближчий до Q логічний попередник, позначимо його R. Літера L вважається *відомостю порядку* 2 про задане входження t, якщо t є одним з аргументів L, і секвенція $\bar{\mathbb{L}}, \mathbb{L}^0, \mathbb{L}^1, |R| \to \langle\!|L|\!\rangle_\pi^{|P|}$ може бути виведена в $\mathbf{S}$. Позначимо множину усіх відомостей порядку 2 через $\mathbb{L}^2$.

Будемо повторювати цей процес, доки не дійдемо до початку тексту. Об'єднані множини відомостей усіх досягнутих порядків становлять сукупну множину $\mathbb{L}$ відомостей про задане входження t.

Розглянемо формульний образ простого твердження: `forall x ((x is a divisor of N) implies ((x + 1) is a natural number))`. Припустимо, що серед логічних попередників цього твердження є декларація змінної `N` як натурального числа, а також визначення поняття `divisor of`. Розглянемо терм `x + 1`. Його підтермами є `x` та `1`; про перший з них маємо такі відомості: `x is a divisor of N` (береться з локального контексту входження) та `x is a natural number` (виводиться за один крок з попереднього факту, декларації змінної `N` та визначення дільника); про другий – `1 is a natural number` (береться з множини логічних попередників речення). Якщо серед логічних попередників також є твердження, що сума двох натуральних чисел є натуральним числом, то атом `(x + 1) is a natural number` буде виведений в один крок і стане відомістю про терм `x + 1`.

Зауважимо, що сукупності відомостей, які пов'язуються з входженнями термів, дають нам дуже просту, коректну, але, звісно, неповну процедуру "швидкої" перевірки атомарних цілей: маємо лише переглянути відомості про аргументи в цілі. Цікаво зазначити, що в накопичені відомості природно потрапляє в першу чергу інформація про типи термів. Це зближає запропонований підхід з методом, викладеним в [47].

2.4. Коректність ForTheL-тексту

Ми розглядаємо два типи коректності ForTheL-тексту: онтологічну коректність, яка виражає "осмисленість" тексту, та логічну коректність, яка виражає його "справедливість", точніше, справедливість зроблених в ньому тверджень. Зауважимо, що логічно коректний текст не обов'язково є онтологічно коректним (таким, наприклад, є ForTheL-текст, наведений в додатку роботи), але загально коректний текст має задовольняти обом умовам.

Неформально, текст є *онтологічно коректним*, якщо, по-перше, кожному аргументному місцю кожного синтаксичного примітива, що вводиться в тексті, співставлено деяке поняття (тип), і, по-друге, будь-який терм, що зустрічається в цьому аргументному місці, належить до об'єму цього поняття. Сукупність типів аргументних місць фактично задає область визначеності відповідної функції, предиката або поняття. Інакше кажучи, в онтологічно коректному тексті всім синтаксичним примітивам може бути надана теоретико-множинна інтерпретація. Поняття онтологічної коректності в мові ForTheL відповідає поняттю типізованості в формалізмі WTT (Weak Type Theory) [48].

Перша умова задовольняється, якщо кожен синтаксичний примітив вводиться або в розділі визначення, або в сигнатурному розділі. За правилами мови ForTheL, типи аргументних місць явно перелічуються в цих розділах. Визначення онтологічної коректності дозволяє певну омонімію: один синтаксичний примітив може бути введений декілька разів, але вимагається, щоб області визначеності таких омонімів не перетиналися: жоден кортеж аргументів не може одночасно задовольняти двом різним наборам типів.

Друга умова перевіряється наступним чином. Для кожного входження деякого предиката, функції або поняття в формульний образ речення в нормалізованому ForTheL-тексті, ми маємо знайти серед логічних попередників цього речення такий розділ визначення або сигнатурний розділ, в якому вводиться відповідний синтаксичний примітив, і необхідні твердження про належність аргументів цього входження до об'ємів понять-

типів є локально істинними в позиції цього входження в формульному образі — тобто локальний образ твердження випливає з множини (формульних образів) логічних попередників.

Перейдемо до точних визначень. Нехай маємо правильно побудований ForTheL-текст $\mathbb{T}$. Розглянемо деяке речення P в нормалізованому тексті $\mathbb{T}^\circ$. Нехай $\pi \in \Pi(|P|)$ є позицією атому чи терму в формульному образі речення P (нагадаємо, що синтаксичні примітиви понять в формульних образах відповідають предикатним символам).

Входження, визначене позицією π, вважається *онтологічно коректним*, або якщо в ньому знаходиться одна з перелічених конструкцій:

- головний юніт визначення чи сигнатурного розділу;
- пропозиційна константа `truth` чи `contradiction`;
- атом рівності $term$ `is equal to` $term$;
- терм-змінна;

або якщо в множині логічних попередників $\mathbf{LP}_{\mathbb{T}^\circ}(P)$ знайдеться *один і лише один* такий розділ D, що виконуються всі наступні умови:

- D є розділом визначення або сигнатурним розділом з початковими припущеннями $A_1, \ldots, A_r$ і твердженням S;
- головний юніт в твердженні S (після розкриття синонімів) утворюється з того самого синтаксичного примітиву, що й входження $(|P|)|_\pi$;
- позначивши терми-аргументи входження $(|P|)|_\pi$ через $s_0, \ldots, s_n$, а змінні-аргументи головного юніту в формульному образі $|S|$ (дивись розділ 1.2.8) — через $v_0, \ldots, v_n$, маємо

$$\mathbf{LP}_{\mathbb{T}^\circ}(P) \models \langle\!|(|A_1| \wedge \cdots \wedge |A_r|)\sigma|\!\rangle_\pi^{|P|}$$

де σ є підстановкою $[s_0/v_0, \ldots, s_n/v_n]$.

Зауважимо, що серед вільних змінних (формульних образов) речень A_1, $\ldots$, A_r немає інших, крім $v_0, \ldots, v_n$ — цього вимагають умови правильно

побудованості тексту, сформульовані в підрозділі 1.3.5. Множина ForTheL-розділів $\mathbf{LP}_{\mathbb{T}^\circ}(P)$ справа від знаку $\models$ розуміється як множина формульних образів цих розділів.

ForTheL-текст $\mathbb{T}$ вважається *онтологічно коректним*, якщо усі входження атомів та термів в формульні образи речень в нормалізованому тексті $\mathbb{T}^\circ$ є онтологічно коректними.

ForTheL-текст $\mathbb{T}$ є *логічно коректним*, якщо кожне твердження або селекція P, що міститься всередині розділу-теореми нормалізованого тексту $\mathbb{T}^\circ$, випливає з множини своїх логічних попередників $\mathbf{LP}_{\mathbb{T}^\circ}(P)$:

- якщо P є твердженням, то $\quad \mathcal{I}nd,\ \mathbf{LP}_{\mathbb{T}^\circ}(P)\ \models\ |P|$
- якщо P є селекцією $(selPrefix\ \ (notions)_N\ .)$, то

$$\mathcal{I}nd,\ \mathbf{LP}_{\mathbb{T}^\circ}(P)\ \models\ |\texttt{there exist}\ N\ .|$$

Тут $\mathcal{I}nd$ позначає схему аксіом загальної індукції для фундованого впорядкування `-<-`:

$$\forall\vec{x}\big(\big(\forall\vec{y}((t[\vec{y}] \prec\!\!\!\prec t[\vec{x}]) \supset F[\vec{y}])\big) \supset F[\vec{x}]\big) \supset \forall\vec{x}F[\vec{x}]$$

Зауважимо, що процедура нормалізації ForTheL-тексту має таку властивість: якщо в нормалізованому тексті деяке твердження супроводжується доведенням, і це доведення є логічно коректним, то й саме це твердження є коректним, тобто випливає з множини своїх логічних попередників. Це трапляється завдяки тому, що в кінець нормалізованого ForTheL-доведення ставиться поточна цільова пропозиція, і, таким чином, формульний образ розділу доведення вже "містить" початкове твердження, яке доводиться. Доведення по індукції, в які додається припущення гіпотези індукції, не спростовують це правило, оскільки в присутності схеми аксіом $\mathcal{I}nd$ це припущення не обмежує загальності.

Таким чином, перевіряючи логічну коректність тексту, ми можемо обмежитись лише тими твердженнями, які не супроводжуються доведеннями.

2.5. Процедури перевірки коректності

З наведених визначень видно, що встановлення коректності (як онтологічної, так і загальної) ForTheL-тексту зводиться до наступного: для кожного речення S в нормалізованому тексті маємо перевірити, що певна сукупність тверджень (цілей) випливає з множини логічних попередників S (контексту). Кожна така перевірка постає окремим завданням для ядра системи САД, так званого "міркувальника".

Для виконання такого завдання в міркувальника є арсенал елементарних дій:

- Направити поточне завдання пруверу, с вказанням ліміту часу на перевірку. Оскільки прувер в системі САД працює з формулами першого порядку, речення та розділи ForTheL замінюються їхними формульними образами. Якщо уявити, що в якості прувера ми маємо несхибного оракула, цієї елементарної дії цілком достатньо, і в міркувальнику, як такому, немає потреби. Втім на практиці, завдання, що постають в нетривіальних математичних текстах, виявляються, у їхньому початковому вигляді, занадто складними навіть для найкращих існуючих пруверів.

- Якщо цілева формула в поточному завданні є літерою L, застосувати "швидку" перевірку, тобто пересвідчитись, чи не міститься L серед відомостей про терми-аргументи цілі.

- Якщо цілева формула є пропозиціональною комбінацією декількох формул або ж формулою під універсальним квантором, застосувати класичні правила Е. Бета [49]. Ця дія зводить початкове завдання до одного чи декількох (можливо) простіших завдань.

- Скоротити контекст завдання, викресливши з нього деякі розділи-попередники.

- Розкрити визначення для деякої сукупності входжень, за умови, що відповідний символ введений у розділі визначення, і онтологічна коректність входження вже перевірена.

- Добавити в контекст завдання висновок деякої леми, після того, як перевірено її умови в поточному контексті.

- Добавити в контекст деякі відомості про терми, що зустрічаються в завданні.

Послідовність цих дій та їхні параметри (входження, де розкриваються визначення; попередники, що видаляються з контексту; вибір лемм, що застосовуються, та інстанціювання універсальних змінних в них) визначаються в залежності від характеру та вигляду завдання за деяким алгоритмом. В перспективі розвитку системи САД цей алгоритм має надавати користувач системи — у вигляді програми, інтерпретатором якої виступатиме міркувальник.

В існуючій версії системи відповідний алгоритм запрограмовано безпосередньо в міркувальнику. Коротко перелічимо деякі його особливості.

- Кожна цільова формула розщеплюється на підформули наскільки можливо.

- Кожна літерна ціль перевіряється за "швидкою" процедурою перед тим, як до цього завдання будуть застосовані ще якісь дії. В разі перевірки онтологічної коректності, для літерних цілей ми обмежуємось "швидкою" процедурою.

- Розділи визначення видаляються з контексту перед тим, як завдання віддається пруверу. Ми розраховуємо, що всі необхідні розкриття будуть зроблені міркувальником, і, у такий спосіб, можемо значно скоротити простір пошуку для прувера.

- Порядок розкриття визначень встановлюється за такими критеріями, як місцезнаходження входження в тексті (символи, які зустрічаються в цілі, варто розкривати раніше, ніж ті, що взагалі не з'являються в "ближньому околі" поточного речення) та ієрархія визначень (якщо символ P визначається за допомогою символу Q, варто розкривати входження P перед входженнями Q).

РОЗДІЛ 3
ЦІЛЕКЕРОВАНИЙ ПОШУК ВИВЕДЕННЯ

В цьому розділі вивчається і набуває подальшого розвитку підхід до автоматичного пошуку виведення, прийнятий у програмі "Алгоритм Очевидності". В попередніх роботах, присвячених цій темі [50, 51, 52], опис дедуктивної процедури у системі САД ґрунтувався на спеціальних численнях секвенціального типу, що мали наступні характерні риси:

- виведення йде у напрямку від цілевої секвенції до аксіом;
- виведення є цілекерованим, тобто наступний крок виведення визначається формою формули-консеквента;
- початкові формули-антецеденти не змінюються, до них додаються породжені атомарні "леми";
- підстановка у вільні змінні визначається за системою уніфікаційних рівнянь, накопичених впродовж виведення;
- початкові формули не сколемізуються, натомість, сукупна підстановка у вільні змінні має задовольняти спеціальній умові допустимості.

Такі властивості системи виведення дозволяють природним чином викласти її у рамках табличного формалізму [53]. Вивідним об'єктом табличного числення є дерево формул, окремий крок виведення полягає у доданні до деякої гілки у цьому дереві нових листів. Виведення починається з єдиної гілки — сукупності формул, чию суперечливість ми намагаємось довести. Виведення вважається завершеним, коли кожна гілка у отриманому дереві містить дві комплементарні літери. Таким чином, цілекеровані секвенціальні числення у стилі АО легко переформулювати як табличні:

- початкова секвенція $F_1, \ldots, F_n \to G$ перетворюється на початкову гілку $F_1, \ldots, F_n, \neg G$;

- один крок виведення у секвенціальному численні — породження однієї чи декількох секвенцій з консеквентами $G_1, \ldots, G_s$ — відповідає одному кроку виведення у табличному численні — додання до деякої гілки листів $G_1^\neg, \ldots, G_s^\neg$, де $G_i^\neg$ є формула, комплементарна до $G_i^\neg$;

- атомарні "леми", накопичені в окремій секвенції, містяться у гілці над вузлом, що відповідає цій секвенції у дереві табличного виведення.

3.1. Загальні означення

Як і в попередньому розділі, ми розглядаємо класичну логіку першого порядку з наступними зв'язками: $\supset$ (імплікація), $\vee$ (диз'юнкція), $\wedge$ (кон'юнкція), $\neg$ (заперечення), $\forall$ (квантор загальності), $\exists$ (квантор існування), $\bot$ (хибність) — але без еквівалентності. Еквівалентність $F \equiv G$ вважається тепер абревіатурою для $(F \supset G) \wedge (G \supset F)$.

Поняття підстановки, субституту, позиції входження в формулі, відповідна нотація та математичні операції також розуміються згідно з визначеннями в підрозділі 2.1.

В наших численнях, ми використовуємо поняття констрейнту. Атомарний констрейнт може мати одну з трьох форм: $t = s$, $t \neq s$, $t < s$, де t та s є термами зі змінними. Семантика нерівності визначається деяким повним фундованим впорядкуванням на основних термах (зауважимо, що в конкретному численні це впорядкування може мати додаткові властивості).

Констрейнтом є кон'юнкція атомарних констрейнтів. Констрейнт вважається *виконуваним*, якщо існує підстановка, яка *задовольняє його*, тобто перетворює усі атомарні констрейнти у ньому в основні рівняння або нерівності. Пустий констрейнт задовольняється будь-якою підстановкою.

Ми приводимо декілька табличних числень у цій главі, тому необхідно прийняти уніфіковане представлення для них. *Табло* — це скінчене дерево $\mathbb{T}$, в вузлах якого (за винятком кореня) стоять пари $F \cdot \gamma$, де F — формула, а γ — констрейнт. Корінь дерева (або *початковий вузол*) містить початкову множину формул, що має бути спростована.

Будь-яка гілка, що містить формулу $\bot$ вважається *закритою*. Табло називається *закритим*, коли кожна гілка у ньому закрита, а совокупна множина констрейнтів є виконуваною.

Виведення починається з початкового вузла. Кожний наступний крок розширює деяку гілку в табло, додаючи до неї знизу нові листи (або піддерева). Правила виведення можуть бути описані наступним чином:

$$\frac{\Gamma \parallel \Delta}{\mathbb{T}_1 \quad \cdots \quad \mathbb{T}_n}$$

де Γ — початкова множина формул (початковий вузол), Δ — гілка, яку ми розширюємо (випускаючи констрейнти), $\mathbb{T}_1, \ldots, \mathbb{T}_n$ — додані піддерева. Порядок формул в Γ та Δ є істотним. Якщо ми не вказуємо констрейнт поруч з формулою у $\mathbb{T}_i$, він вважається пустим.

Як зразок, розглянемо формулювання класичного табличного числення першого порядку **Tb** на рисунку 3. На цьому рисунку (та надалі) $\mathcal{FV}(F)$ позначає список вільних змінних у формулі F, а символи з рисками $\bar{u}, \bar{\mathbf{g}}$ вважаються новими по відношенню до табло в цілому. Таким чином, γ-правила підставляють нову змінну замість зв'язаної, а δ-правила вводять новий сколемівський символ.

Як відомо, числення **Tb** є коректним і повним у класичній логиці першого порядку:

Факт 3.1. *Скінчена множина замкнених формул є несумісною тоді і тільки тоді, коли з неї може бути виведено закрите табло в* **Tb**.

3.2. Цілекероване табличне числення

3.2.1. Допустимі підстановки

Поняття допустимої підстановки було введено О.В. Лялецьким. Воно дозволяє забезпечити коректність числень, що застосовують уніфікацію, без використання сколемізації, а отже, зберігаючи початкову сигнатуру задачи. Для числень, що спираються на поняття допустимої підстановки, ми будемо використовувати спеціальний спосіб альфа-конверсії, дещо технічніший, ніж в **Tb**.

Розширення: α-правила:

$$\frac{\Gamma, F, \Lambda \parallel \Delta}{F} \qquad \frac{\Gamma \parallel \Delta, F \wedge G, \Lambda}{\begin{array}{c} F \\ G \end{array}} \qquad \frac{\Gamma \parallel \Delta, \neg(F \vee G), \Lambda}{\begin{array}{c} \neg F \\ \neg G \end{array}} \qquad \frac{\Gamma \parallel \Delta, \neg(F \supset G), \Lambda}{\begin{array}{c} F \\ \neg G \end{array}}$$

Подвійне заперечення: β-правила:

$$\frac{\Gamma \parallel \Delta, \neg\neg F, \Lambda}{F} \qquad \frac{\Gamma \parallel \Delta, \neg(F \wedge G), \Lambda}{\neg F \qquad \neg G} \qquad \frac{\Gamma \parallel \Delta, F \vee G, \Lambda}{F \qquad G} \qquad \frac{\Gamma \parallel \Delta, F \supset G, \Lambda}{\neg F \qquad G}$$

γ-правила: δ-правила:

$$\frac{\Gamma \parallel \Delta, \forall\, v\, F, \Lambda}{F[\bar{u}/v]} \qquad \frac{\Gamma \parallel \Delta, \neg\exists\, v\, F, \Lambda}{\neg F[\bar{u}/v]} \qquad \frac{\Gamma \parallel \Delta, \exists\, v\, F, \Lambda}{F[\bar{\mathbf{g}}(\mathcal{FV}(F))/v]} \qquad \frac{\Gamma \parallel \Delta, \neg\,\forall\, v\, F, \Lambda}{\neg F[\bar{\mathbf{g}}(\mathcal{FV}(F))/v]}$$

Термінальні правила

$$\frac{\Gamma \parallel \Delta, \neg P(s_1, \ldots, s_n), \Lambda, P(t_1, \ldots, t_n)}{\bot \cdot (s_1 = t_1 \wedge \cdots \wedge s_n = t_n)} \qquad \frac{\Gamma \parallel \Delta, P(s_1, \ldots, s_n), \Lambda, \neg P(t_1, \ldots, t_n)}{\bot \cdot (s_1 = t_1 \wedge \cdots \wedge s_n = t_n)}$$

Рис. 3: Просте табличне числення **Tb**

Скінчена множина замкнених формул Γ називається *безконфліктною*, якщо в ній немає двох кванторів по одній змінній. Позначимо через $\mathcal{V}_\Gamma$ множину *індексованих змінних* $\{\, {}^k v \,|\, v \text{ occurs in } \Gamma,\ k \in \mathbf{N} \,\}$. Вираз ${}^k F$ позначатиме формулу F, де кожна змінна v, вільна чи зв'язана, замінена на індексовану змінну ${}^k v$. Зауважимо, що v, ${}^k v$ та ${}^{k+1} v$ — це три різні змінні. Γ-*термом* будемо називати терм зі змінними з $\mathcal{V}_\Gamma$ та функціональними символами з Γ.

Будемо називати змінну ${}^k v \in \mathcal{V}_\Gamma$ *невідомою* в Γ (${}^k v \in \mathcal{V}_\Gamma^+$), якщо деяка формула в Γ має вигляд $P[(\forall v F)^+]_\tau$ або $P[(\exists v F)^-]_\tau$. Відповідно, ${}^k v \in \mathcal{V}_\Gamma$ назвемо *фіксованою* в Γ (${}^k v \in \mathcal{V}_\Gamma^-$), якщо Γ містить формулу вигляду $P[(\forall v F)^-]_\tau$ або $P[(\exists v F)^+]_\tau$. Очевидно, кожна змінна в $\mathcal{V}_\Gamma$ є або невідомою, або фіксованою: $\mathcal{V}_\Gamma^+ \cup \mathcal{V}_\Gamma^- = \mathcal{V}_\Gamma$, $\mathcal{V}_\Gamma^+ \cap \mathcal{V}_\Gamma^- = \varnothing$. В подальшому, ми будемо записувати фіксовані змінни полужирним шрифтом: ${}^k \mathbf{v}$.

Безконфліктна множина формул Γ породжує відношення $\lhd_\Gamma$: $\mathcal{V}_\Gamma^+ \times \mathcal{V}_\Gamma^-$ наступним чином: ${}^k u \lhd_\Gamma {}^m \mathbf{w}$ тоді і тільки тоді, коли $k = m$ та квантор по w входить в область дії квантора по u в Γ, інакше кажучи, коли деяка

формула $F \in \Gamma$ має вигляд $(\dots \mathcal{Q}_1\, u\, (\dots \mathcal{Q}_2\, w\, (\dots)\dots)\dots)$. Очевидно, з ${}^k u \lhd_\Gamma {}^k\mathbf{w}$ випливає ${}^m u \lhd_\Gamma {}^m\mathbf{w}$.

Для заданої підстановки σ, ми вводимо відношення $\lll^\sigma_\Gamma$: $\mathcal{V}^-_\Gamma \times \mathcal{V}^+_\Gamma$ так: ${}^m\mathbf{w} \lll^\sigma_\Gamma {}^k u$ тоді і лише тоді, коли ${}^m\mathbf{w}$ входить в ${}^k u\sigma$.

Скінчена підстановка σ є *допустимою* в Γ, якщо виконуються наступні умови:

1. для будь-якої фіксованої ${}^k\mathbf{w} \in \mathcal{V}^-_\Gamma$, існує деякий індекс m, такий що ${}^k\mathbf{w}\sigma = {}^m\mathbf{w}$;

2. для будь-яких ${}^k\mathbf{w}, {}^m\mathbf{w} \in \mathcal{V}^-_\Gamma$, ${}^k u \in \mathcal{V}^+_\Gamma$, якщо ${}^k\mathbf{w}\sigma = {}^m\mathbf{w}\sigma$ та ${}^k u \lhd_\Gamma {}^k\mathbf{w}$, то ${}^k u\sigma = {}^m u\sigma$;

3. транзитивне замикання композиції $(\lll^\sigma_\Gamma \circ \lhd_\Gamma)$ іррефлексивне.

Наступний аналог теореми Ербрана має місце для допустимих підстановок:

Теорема 3.2. *Нехай Γ є безконфліктною множиною замкнених формул. Позначимо через Γ' множину формул з Γ з викресленими кванторами (тобто, всі змінні в Γ' вільні). Множина Γ є сумісною тоді і лише тоді, коли для всіх $n \in \mathbf{N}$ та будь-якої скінченої допустимої підстановки σ в сигнатурі Γ, множина $\{{}^1\Gamma', \dots, {}^n\Gamma'\}\sigma$ є сумісною.*

Нижче ми доведемо цей результат в іншому аспекті, встановивши зв'язок між поняттям допустимої підстановки та сколемізацією.

Сколемізуюча підстановка θ_Γ заміщає кожну фіксовану змінну ${}^k\mathbf{v}$ сколемівським функціональним символом $\mathbf{v}({}^k u_1, \dots, {}^k u_n)$, де ${}^k u_1, \dots, {}^k u_n$ є максимальною множиною невідомих змінних, таких що ${}^k u_i \lhd_\Gamma {}^k\mathbf{v}$ і квантор по u_{i+1} входить в області дії квантору по u_i в Γ. Зауважимо, що θ_Γ є необхідно нескінченою.

Теорема 3.3. *Нехай Γ є безконфліктною множиною замкнених формул, а σ — допустимою підстановкою в Γ. Існує підстановка π, така що для будь-яких Γ-термів s, t, з $s\sigma = t\sigma$ випливає $s\theta_\Gamma\pi = t\theta_\Gamma\pi$.*

Доведення. Ми можемо припустити, що будь-яка змінна, невідома чи фіксована, в яку σ робить підстановку, не зустрічається в термах-субститутах σ (кожна така змінна може бути переіменована в субституті).

Позначимо через ϕ підстановку $(\theta_\Gamma)\sigma$, тобто підстановку, яка замінює будь-яку фіксовану змінну ${}^i\mathbf{v}$ на терм ${}^i\mathbf{v}\theta_\Gamma\sigma$. Зауважимо, що ϕ не є композицією θ_Γ з σ; наприклад, ϕ не робить підстановку в невідомі змінні. Підстановка ϕ утворюється застосуванням σ до субститутів θ_Γ.

Нехай ϕ^1 позначає ϕ, ϕ^2 позначає $\phi \circ \phi$, а ϕ^{i+1} позначає $\phi^i \circ \phi$ (тут, $\circ$ позначає композицію підстановок). Доведемо, що існує таке n, що $\phi^{n+1} = \phi^n$. Позначимо через W_i множину фіксованих змінних, що входять в субститути ϕ^i. Зауважимо, що W_i скінчена та $W_{i+1} \subseteq W_i$ для всіх $i > 0$. Оскільки транзитивне замикання $(\lessdot\!\lhd_\Gamma^\sigma \circ \lhd_\Gamma)$ іррефлексивне, тобто є строгим частковим впорядкуванням, ми можемо вибрати в W_i максимальний елемент по відношенню к цьому впорядкуванню, за умови, що W_i непусте. Позначимо цей елемент ${}^k\mathbf{w}$. Тепер, якщо знайдется така змінна ${}^l\mathbf{v} \in W$, що ${}^l\mathbf{v}\phi$ містить ${}^k\mathbf{w}$, то матимемо ${}^k\mathbf{w}\,(\lessdot\!\lhd_\Gamma^\sigma \circ \lhd_\Gamma)\,{}^l\mathbf{v}$, всупереч максимальності ${}^k\mathbf{w}$. Справді, якщо ${}^l\mathbf{v}\phi$ містить ${}^k\mathbf{w}$, тоді для деякого ${}^l u \lhd_\Gamma {}^l\mathbf{v}$, ${}^l u\sigma$ містить ${}^k\mathbf{w}$, а отже ${}^k\mathbf{w} \lessdot\!\lhd_\Gamma^\sigma {}^l u$. Таким чином, якщо W_i непусте, то W_{i+1} є власною підмножиною W_i. Тому має існувати натуральне n таке що $W_n = \varnothing$. Очевидно, $\phi^n \circ \phi = \phi^n$.

Нехай ψ позначає ϕ^n, а π позначає $\sigma \circ \psi$. Для будь-якої фіксованої змінної ${}^k\mathbf{v}$, ${}^k\mathbf{v}\theta_\Gamma\sigma = {}^k\mathbf{v}\sigma\theta_\Gamma\sigma$, за визначенням допустимості. Тоді маємо ${}^k\mathbf{v}\theta_\Gamma\pi = {}^k\mathbf{v}\theta_\Gamma\sigma\psi = {}^k\mathbf{v}\sigma\theta_\Gamma\sigma\psi = {}^k\mathbf{v}\sigma\phi\psi = {}^k\mathbf{v}\sigma\psi$. Для будь-якої невідомої змінної ${}^k u$, теж маємо ${}^k u\theta_\Gamma\pi = {}^k u\sigma\psi$. Отже для будь-якого Γ-терму r, $r\theta_\Gamma\pi = r\sigma\psi$.

Очевидно для всіх s та t, $s\sigma = t\sigma \;\Rightarrow\; s\sigma\psi = t\sigma\psi \;\Rightarrow\; s\theta_\Gamma\pi = t\theta_\Gamma\pi$. □

Теорема 3.4. *Нехай* Γ *є безконфліктною множиною замкнених формул, а* π *— підстановкою. Існує підстановка* σ *допустима в* Γ*, така що для будь-яких* Γ*-термів* s*,* t*, з* $s\theta_\Gamma\pi = t\theta_\Gamma\pi$ *випливає* $s\sigma = t\sigma$*.*

Доведення. Ми можемо припустити, що π не підставляє в фіксовані змінні і не містить фіксованих змінних у своїх термах-субститутах. Ми мо-

жемо також припустити, що будь-яка невідома змінна, в яку π робить підстановку, не зустрічається в субститутах π.

Позначимо через ψ підстановку $(\theta_\Gamma)\pi$, тобто підстановку, що заміщує кожну фіксовану змінну ${}^i\mathbf{v}$ на терм ${}^i\mathbf{v}\theta_\Gamma\pi$. Зауважимо, що ψ не є композицією θ_Γ з π; наприклад, ψ не робить підстановки в невідомі змінні. Підстановка ψ утворюється застосуванням π до всіх субститутів θ_Γ.

Введемо відношення еквівалентності $\bowtie$ на фіксованих змінних з $\mathcal{V}_\Gamma$ наступним чином: ${}^k\mathbf{w} \bowtie {}^l\mathbf{v}$ тоді і лише тоді, коли ${}^k\mathbf{w}\psi = {}^l\mathbf{v}\psi$. Інакше кажучи, ${}^k\mathbf{w} \bowtie {}^l\mathbf{v}$, коли w та v є однією й тією самою змінною в Γ, і, для всіх ${}^k u \lhd_\Gamma {}^k\mathbf{w}$, ${}^k u\pi = {}^l u\pi$. Оскільки π скінчена, фактор-множина $\mathcal{V}_\Gamma^- / \bowtie$ матиме скінчену кількість членів з двома чи більше елементами.

Підстановка σ визначається наступним чином. Для кожної фіксованої змінної ${}^l\mathbf{v}$, ${}^l\mathbf{v}\sigma = {}^k\mathbf{v}$, де k — це найменший індекс в класі еквівалентності ${}^l\mathbf{v}$. Для кожної невідомої змінної ${}^l u$, ${}^l u\sigma$ є термом ${}^l u\pi$, де певні підтерми замінені фіксованими змінними за наступним правилом: підтерм r терму ${}^l u\pi$ замінено на фіксовану змінну ${}^k\mathbf{v}$ в ${}^l u\sigma$, якщо $r = {}^k\mathbf{v}\psi$, жоден сколемівський символ не входить над r в ${}^l u\pi$, а k є найменшим індексом в класі еквівалентності ${}^k\mathbf{v}$.

Легко побачити, що σ допустима в Γ. Перша та друга умова в визначенні допустимості з очевидністю мають місце. Для того, щоб показати, що третя умова також задоволена, зауважимо, що з ${}^k u\,(\lhd_\Gamma \circ \lll_\Gamma^\sigma)\,{}^l x$ випливає, що ${}^k u\pi$ є власним підтермом ${}^l x\pi$. Транзитивне замикання $(\lhd_\Gamma \circ \lll_\Gamma^\sigma)$ є, таким чином, іррефлексивним, а отже транзитивне замикання $(\lll_\Gamma^\sigma \circ \lhd_\Gamma)$ також іррефлексивне.

Нехай k буде найменшим індексом в класі еквівалентности фіксованої змінної ${}^l\mathbf{v}$. Ми отримуємо ${}^l\mathbf{v}\theta_\Gamma\pi = \mathbf{v}({}^l u\pi, \dots) = \mathbf{v}({}^k u\pi, \dots) = {}^k\mathbf{v}\theta_\Gamma\pi = {}^k\mathbf{v}\psi = {}^l\mathbf{v}\sigma\psi$. Для будь-якої невідомої змінної ${}^l u$, отримуємо ${}^l u\theta_\Gamma\pi = {}^l u\pi = {}^l u\sigma\psi$ за побудовою σ. Отже, для будь-якого Γ-терму r, $r\theta_\Gamma\pi = r\sigma\psi$.

Розглянемо Γ-терми s, t такі, що $s\theta_\Gamma\pi = t\theta_\Gamma\pi$. За попереднім, маємо $s\sigma\psi = t\sigma\psi$. Припустимо, що $s\sigma \neq t\sigma$. Оскільки ψ підставляє лише в фіксовані змінні, можемо припустити, що існує підтерм r терму $s\sigma$ і фіксована змінна ${}^k\mathbf{v}$ в $t\sigma$ такі, що $r\psi = {}^k\mathbf{v}\psi$, але $r \neq {}^k\mathbf{v}$.

За побудовою σ, змінна ${}^k\mathbf{v}$ повинна мати найменший індекс в своєму класі еквівалентності. Якщо r є фіксованою змінною, вона також має найменший індекс в своєму класі. Оскільки $r \bowtie {}^k\mathbf{v}$, отримуємо $r = {}^k\mathbf{v}$. В іншому випадку, r є терм з головним сколемівським символом $\mathbf{v}$. Оскільки терм s не містить сколемівських символів, r мав потрапити в $s\sigma$ в субституті σ. Отже, $r\psi$ входить в деякий субститут π. Але $r\psi = {}^k\mathbf{v}\psi$, отже r не може входити в субститут σ, за побудовою σ. Ми прийшли до протиріччя. Отже $s\sigma = t\sigma$. □

3.2.2. Цілекероване табличне числення

Базове числення прувера системи САД є варіантом числення **Tb**, де вибір чергового кроку виведення для даної гілки керується формою листа гілки, який ми називаємо *ціллю*: поки в листі стоїть складена нелітерна формула, ми застосовуємо $\alpha\beta\gamma\delta$-правила, що розбивають ціль; коли в листі опиняється літера, обираємо нову ціль.

Вибір наступного кроку можна додатково скоротити, зафіксувавши деякий атом в цілевій формулі. Інакше кажучи, на кожному кроці виведення, в кожному новопородженому листі є виділений атом. Для цього, ми працюватимемо з деревами трійок $F \diamond \tau \cdot \gamma$, де F є формула, γ — констрейнт, а τ — позиція (виділеного атому) в F.

На початку виведення, ми обираємо одну з початкових формул, виділяємо в ній деякий атом і ставимо її в лист під кореневим вузлом.

Розбиваючи ціль, ми зберігаємо виділений атом. В α-правилах ми додаємо до гілки лише одну з двох підформул: ту, що містить виділений атом. В β-правилах одна з породжених гілок успадковує виділений атом, а в другій ми виберемо новий виділений атом довільним чином.

Якщо цілева формула є літерою L (з виділеним атомом A), і правила розбиття незастосовні, вибір нової цілі також може бути цілекерованим: серед початкових формул ми виберемо ту, що містить входження атома, яке можна уніфікувати з A, за умови, що знак входження протилежний знаку входження A в L.

Оскільки ми працюємо з допустимими підстановками, і допустимість

Правило старту:

$$\frac{\Gamma,\ F[P(\vec{s})]_\tau \parallel}{{}^0F[P(\vec{s})]_\tau \diamond \tau}$$

Подвійне заперечення:

$$\frac{\Gamma \parallel \Delta,\ \neg\neg F \diamond 0.0.\tau}{F \diamond \tau}$$

α-правила:

$$\frac{\Gamma \parallel \Delta,\ F \wedge G \diamond 0.\tau}{F \diamond \tau} \qquad \frac{\Gamma \parallel \Delta,\ F \wedge G \diamond 1.\tau}{G \diamond \tau}$$

$$\frac{\Gamma \parallel \Delta,\ \neg(F \vee G) \diamond 0.0.\tau}{\neg F \diamond 0.\tau} \qquad \frac{\Gamma \parallel \Delta,\ \neg(F \vee G) \diamond 0.1.\tau}{\neg G \diamond 0.\tau}$$

$$\frac{\Gamma \parallel \Delta,\ \neg(F \supset G) \diamond 0.0.\tau}{F \diamond \tau} \qquad \frac{\Gamma \parallel \Delta,\ \neg(F \supset G) \diamond 0.1.\tau}{\neg G \diamond 0.\tau}$$

β-правила:

$$\frac{\Gamma \parallel \Delta,\ \neg(F \wedge G[Q(\vec{s})]_\mu) \diamond 0.0.\tau}{\neg F \diamond 0.\tau \qquad \neg G[Q(\vec{s})]_\mu \diamond 0.\mu} \qquad \frac{\Gamma \parallel \Delta,\ \neg(F[Q(\vec{s})]_\mu \wedge G) \diamond 0.1.\tau}{\neg F[Q(\vec{s})]_\mu \diamond 0.\mu \qquad \neg G \diamond 0.\tau}$$

$$\frac{\Gamma \parallel \Delta,\ F \vee G[Q(\vec{s})]_\mu \diamond 0.\tau}{F \diamond \tau \qquad G[Q(\vec{s})]_\mu \diamond \mu} \qquad \frac{\Gamma \parallel \Delta,\ F[Q(\vec{s})]_\mu \vee G \diamond 1.\tau}{F[Q(\vec{s})]_\mu \diamond \mu \qquad G \diamond \tau}$$

$$\frac{\Gamma \parallel \Delta,\ F \supset G[Q(\vec{s})]_\mu \diamond 0.\tau}{\neg F \diamond 0.\tau \qquad G[Q(\vec{s})]_\mu \diamond \mu} \qquad \frac{\Gamma \parallel \Delta,\ F[Q(\vec{s})]_\mu \supset G \diamond 1.\tau}{\neg F[Q(\vec{s})]_\mu \diamond 0.\mu \qquad G \diamond \tau}$$

$\gamma\delta$-правила:

$$\frac{\Gamma \parallel \Delta,\ \forall^k v\ F \diamond 0.\tau}{F \diamond \tau} \qquad \frac{\Gamma \parallel \Delta,\ \neg\exists^k v\ F \diamond 0.0.\tau}{\neg F \diamond 0.\tau}$$

$$\frac{\Gamma \parallel \Delta,\ \exists^k v\ F \diamond 0.\tau}{F \diamond \tau} \qquad \frac{\Gamma \parallel \Delta,\ \neg\forall^k v\ F \diamond 0.0.\tau}{\neg F \diamond 0.\tau}$$

Вибір нової цілі (m є новим індексом):

$$\frac{\Gamma,\ F[P(\vec{s})^-]_\tau,\ \Lambda \parallel \Delta,\ P(\vec{r}) \diamond \varepsilon}{{}^mF[\neg\bot]_\tau \diamond \tau.0 \cdot ({}^ms_1 = r_1 \wedge \cdots \wedge {}^ms_n = r_n)} \qquad \frac{\Gamma,\ F[P(\vec{s})^+]_\tau,\ \Lambda \parallel \Delta,\ \neg P(\vec{r}) \diamond 0}{{}^mF[\bot]_\tau \diamond \tau \cdot ({}^ms_1 = r_1 \wedge \cdots \wedge {}^ms_n = r_n)}$$

Термінальні правила:

$$\frac{\Gamma \parallel \Delta,\ \neg P(\vec{s}) \diamond 0,\ \Lambda,\ P(\vec{r}) \diamond \varepsilon}{\bot \diamond \varepsilon \cdot (s_1 = r_1 \wedge \cdots \wedge s_n = r_n)} \qquad \frac{\Gamma \parallel \Delta,\ P(\vec{s}) \diamond \varepsilon,\ \Lambda,\ \neg P(\vec{r}) \diamond 0}{\bot \diamond \varepsilon \cdot (s_1 = r_1 \wedge \cdots \wedge s_n = r_n)}$$

Рис. 4: Цілекероване табличне числення **GDT**

визначена відносно початкової множини формул, ми можемо не зважати на квантори протягом виведення. Кожного разу, як ми обираємо нову ціль, ми приписуємо всім змінним в ній (як вільним, так і зв'язаним) деякий новий індекс.

Правила виведення цілекерованого табличного числення **GDT** наведені на рисунку 4. Зауважимо, що порядок початкових формул в Γ має значення в **GDT**: правило старту бере останню формулу зі списку. Як і раніше, ми не вказуємо констрейнтів в антецедентах правил — вони можуть бути якими завгодно. В консеквенті правила пропущений констрейнт вважається пустим.

GDT-табло $\mathbb{T}$ з початковим вузлом Γ вважається *закритим*, коли всі гілки в $\mathbb{T}$ закриті і сукупна множина констрейнтів з $\mathbb{T}$ може бути задовільнена Γ-допустимою підстановкою σ.

Числення **GDT** є коректним і повним в логіці першого порядку:

Теорема 3.5. *Нехай* $\Gamma = \Delta \cup \{G\}$ *є безконфліктною множиною замкнених формул. Припустимо, що* Δ *є сумісною. Тоді* Γ *суперечлива тоді і лише тоді, коли з початкового вузла* (Δ, G) *можна побудувати закрите* **GDT**-*табло.*

Числення **GDT** є табличним переформулюванням числення **GD**, чия коректність і повнота були продемонстровані в попередніх роботах за програмою "Алгоритм Очевидності". В підрозділі 3.2.4, ми наводимо непряме доведення теореми 3.5.

Розглянемо приклад виведення в **GDT**. Спростуємо множину Γ:

$$\forall\, pqx\,.\,(p \subseteq q \wedge x \in p) \supset x \in q$$
$$\forall\, ry\,.\,E(r) \supset \neg(y \in r)$$
$$\forall\, s\,.\,(\forall\, z\,.\,\neg(z \in s)) \supset E(s)$$
$$\exists\, e\,.\,E(e)$$
$$\neg\forall\, a\,.\,(\forall\, b\,.\,a \subseteq b) \supset E(a)$$

Зауважимо, що Γ є безконфліктною множиною, отже умови теорем 3.2 та 3.5 виконані і застосування $\gamma\delta$-правил не призведе до колізії імен.

$$
\mathbb{T} = \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\Gamma}{\neg\forall^0\mathbf{a}\,.\,(\forall^0 b\,.\,{}^0\mathbf{a}\subseteq{}^0 b)\supset\boxed{E({}^0\mathbf{a})}\ \diamond\ 0.0.1}\,(\text{start})}{\neg\left((\forall^0 b\,.\,{}^0\mathbf{a}\subseteq{}^0 b)\supset\boxed{E({}^0\mathbf{a})}\right)\ \diamond\ 0.1}\,\delta}{\neg\boxed{E({}^0\mathbf{a})}\ \diamond\ 0}\,\alpha}{\forall^1 s\,.\,(\forall^1\mathbf{z}\,.\,\neg({}^1\mathbf{z}\in{}^1 s))\supset\boxed{\bot}\ \diamond\ 0.1\ \cdot\ ({}^0\mathbf{a}={}^1 s)}\,(\text{ngs})}{(\forall^1\mathbf{z}\,.\,\neg({}^1\mathbf{z}\in{}^1 s))\supset\boxed{\bot}\ \diamond\ 1}\,\gamma}{\neg\forall^1\mathbf{z}\,.\,\neg\boxed{{}^1\mathbf{z}\in{}^1 s}\ \diamond\ 0.0.0 \qquad \boxed{\bot}\ \diamond\ \varepsilon}\,\beta}{\neg\neg\boxed{{}^1\mathbf{z}\in{}^1 s}\ \diamond\ 0.0}\,\delta}{\boxed{{}^1\mathbf{z}\in{}^1 s}\ \diamond\ \varepsilon}\,(\text{dneg})}{\mathbb{T}_1}\,(\text{ngs})
$$

$$
\mathbb{T}_1 = \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\forall^2 p^2 q^2 x\,.\,({}^2 p\subseteq{}^2 q\wedge\neg\boxed{\bot})\supset{}^2 x\in{}^2 q\ \diamond\ 0.0.0.0.1.0\ \cdot\ ({}^1\mathbf{z}={}^2 x\wedge{}^1 s={}^2 p)}{(^2 p\subseteq{}^2 q\wedge\neg\boxed{\bot})\supset{}^2 x\in{}^2 q\ \diamond\ 0.1.0}\,\gamma}{\neg(^2 p\subseteq{}^2 q\wedge\neg\boxed{\bot})\ \diamond\ 0.1.0 \qquad \boxed{{}^2 x\in{}^2 q}\ \diamond\ \varepsilon}\,\beta}{\neg\boxed{{}^2 p\subseteq{}^2 q}\ \diamond\ 0}\,(\text{ngs}) \qquad \dfrac{\mathbb{T}_3}{}\,(\text{ngs})}{\mathbb{T}_2} \qquad \dfrac{\neg\neg\boxed{\bot}\ \diamond\ 0.0}{\boxed{\bot}\ \diamond\ \varepsilon}\,(\text{dneg})
$$

$$
\mathbb{T}_2 = \dfrac{\dfrac{\dfrac{\neg\forall^3\mathbf{a}\,.\,(\forall^3 b\,.\,\boxed{\bot})\supset E({}^3\mathbf{a})\ \diamond\ 0.0.0.0\ \cdot\ ({}^2 p={}^3\mathbf{a}\wedge{}^2 q={}^3 b)}{\neg\left((\forall^3 b\,.\,\boxed{\bot})\supset E({}^3\mathbf{a})\right)\ \diamond\ 0.0.0}\,\delta}{\forall^3 b\,.\,\boxed{\bot}\ \diamond\ 0}\,\alpha}{\boxed{\bot}\ \diamond\ \varepsilon}\,\gamma
$$

$$
\mathbb{T}_3 = \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\forall^4 r^4 y\,.\,E({}^4 r)\supset\neg\neg\boxed{\bot}\ \diamond\ 0.0.1.0.0\ \cdot\ ({}^2 x={}^4 y\wedge{}^2 q={}^4 r)}{E({}^4 r)\supset\neg\neg\boxed{\bot}\ \diamond\ 1.0.0}\,\gamma}{\neg\boxed{E({}^4 r)}\ \diamond\ 0 \qquad \neg\neg\boxed{\bot}\ \diamond\ 0.0}\,\beta}{\exists^5\mathbf{e}\,.\,\boxed{\bot}\ \diamond\ 0\ \cdot\ ({}^4 r={}^5\mathbf{e})}\,(\text{ngs}) \qquad \dfrac{}{\boxed{\bot}\ \diamond\ \varepsilon}\,(\text{dneg})}{\boxed{\bot}\ \diamond\ \varepsilon}\,\delta
$$

Рис. 5: Закрите **GDT**-табло

На рисунку 5 наведено **GDT**-спростування $\mathbb{T}$ множини Γ. Для спрощення подання, виділені атоми обведено. Звернемо увагу на те, що всі гілки в $\mathbb{T}$ закриті, хоча ми жодного разу не застосували термінальне правило: всі потрібні входження $\bot$ були введені правилами вибору нової цілі. Такою є сукупна множина констрейнтів в $\mathbb{T}$:

$$
\begin{array}{llll}
{}^0\mathbf{a} = {}^1s & {}^1\mathbf{z} = {}^2x & {}^1s = {}^2p & {}^2p = {}^3\mathbf{a} \\
{}^2q = {}^3b & {}^2x = {}^4y & {}^2q = {}^4r & {}^4r = {}^5\mathbf{e}
\end{array}
$$

Вона може бути задовільнена підстановкою

$$\sigma = [{}^0\mathbf{a}/{}^3\mathbf{a}, {}^0\mathbf{a}/{}^1s, {}^0\mathbf{a}/{}^2p, {}^1\mathbf{z}/{}^2x, {}^1\mathbf{z}/{}^4y, {}^5\mathbf{e}/{}^2q, {}^5\mathbf{e}/{}^3b, {}^5\mathbf{e}/{}^4r]$$

Для того, щоб пересвідчитись в допустимості σ, розглянемо впорядкування $\lhd_\Gamma$ та $\lll^\sigma_\Gamma$:

$${}^0\mathbf{a} \lll^\sigma_\Gamma {}^1s, {}^2p \quad {}^1\mathbf{z} \lll^\sigma_\Gamma {}^2x, {}^4y \quad {}^5\mathbf{e} \lll^\sigma_\Gamma {}^2q, {}^3b, {}^4r \quad {}^is \lhd_\Gamma {}^i\mathbf{z} \quad (\text{for any } i \in \mathbb{N})$$

Транзитивне замикання $(\lll^\sigma_\Gamma \circ \lhd_\Gamma)$ очевидно не містить циклів. Таким чином, множина Γ успішно спростована.

В наступних підрозділах ми встановимо зв'язок між числення **GDT** та традиційним цілекерованим диз'юнктним табличним численням.

3.2.3. Диз'юнктне цілекероване числення

Числення Clause Connection Tableau, також відоме як Model Elimination, було введено Д. Лавлендом як числення резолюційного типу (з диз'юнктами спеціальної форми) [54]. Піздніше воно було переформульоване як диз'юнктне табличне числення [55], де пошук виведення керується "сполуками" між диз'юнктами: парами протилежних за знаком літер, що можуть бути уніфіковані.

Диз'юнктне табло є скінченим деревом $\mathbb{T}$, де в вузлах, крім кореневого, знаходяться пари $L \cdot \gamma$, де L — літера, а γ — констрейнт. В корені дерева (в початковому вузлі) міститься початкова множина диз'юнктів. Правила виведення числення **CT** наведені на рисунку 6.

Правило старту:

$$\frac{\Gamma,\ (L_1 \vee \cdots \vee L_k),\ \Lambda \parallel}{{}^0L_1 \quad \cdots \quad {}^0L_k}$$

Правила розширення (m є новим індексом):

$$\frac{\Gamma,\ (L_1 \vee \cdots \vee \neg P(\vec{s}) \vee \cdots \vee L_k),\ \Lambda \parallel \Delta, P(\vec{r})}{{}^mL_1 \quad \cdots \quad \bot \cdot ({}^ms_1 = r_1 \wedge \cdots \wedge {}^ms_n = r_n) \quad \cdots \quad {}^mL_k}$$

$$\frac{\Gamma,\ (L_1 \vee \cdots \vee P(\vec{s}) \vee \cdots \vee L_k),\ \Lambda \parallel \Delta, \neg P(\vec{r})}{{}^mL_1 \quad \cdots \quad \bot \cdot ({}^ms_1 = r_1 \wedge \cdots \wedge {}^ms_n = r_n) \quad \cdots \quad {}^mL_k}$$

Термінальні правила:

$$\frac{\Gamma \parallel \Delta, \neg P(\vec{s}), \Lambda, P(\vec{r})}{\bot \cdot (s_1 = r_1 \wedge \cdots \wedge s_n = r_n)} \qquad \frac{\Gamma \parallel \Delta, P(\vec{s}), \Lambda, \neg P(\vec{r})}{\bot \cdot (s_1 = r_1 \wedge \cdots \wedge s_n = r_n)}$$

Рис. 6: Диз'юнктне цілекероване числення **CT**

Факт 3.6. *Множина диз'юнктів $\mathcal{S}$ є суперечливою тоді і тільки тоді, коли з $\mathcal{S}$ можна побудувати закрите* **CT***-табло.*

Властивість повноти може бути посилена:

Теорема 3.7. *Якщо множинв диз'юнктів $\mathcal{S}$ є суперечливою, але будь-яка власна підмножина $\mathcal{S}$ є сумісною, тоді закрите дерево спростування $\mathcal{S}$ в* **CT** *може починатися з будь-якого диз'юнкта $C \in \mathcal{S}$.*

Доведення. Числення **CT** є коректним, отже кожен диз'юнкт з $\mathcal{S}$ має брати участь в будь-якому **CT**-виведенні, що спростовує $\mathcal{S}$, або як стартовий диз'юнкт, або в кроці розширення. Розглянемо закрите табло $\mathbb{T}$, де диз'юнкт C "бере участь" на мінімально можливій глибині (в подальшому будемо називати її "глибиною C"):

$$\frac{\mathcal{S}}{\dfrac{{}^0L_1}{\mathbb{T}_1} \quad \cdots \quad \dfrac{{}^0L_k}{\mathbb{T}_k}}$$

Зауважимо, що $\mathbb{T}_1, \ldots, \mathbb{T}_k$ позначають тут не окремі дерева літер, а набори дерев, оскільки кожний вузол ${}^0L_i$ може мати декілька нащадків. Те ж саме справедливе для $\mathbb{T}_{i1}, \ldots, \mathbb{T}_{il}$ нижче.

Якщо диз'юнкт C є стартовим диз'юнктом $L_1 \vee \cdots \vee L_k$, ми маємо твердження теореми. В противному випадку, C входить в деяке $\mathbb{T}_i$. Розглянемо дерево $\mathbb{T}$ більш детально:

$$\dfrac{\mathcal{S}}{\dfrac{{}^0L_1}{\mathbb{T}_1} \quad \cdots \quad \dfrac{{}^0L_i}{\dfrac{{}^mM_1}{\mathbb{T}_{i1}} \quad \cdots \quad \perp \cdot \gamma \quad \cdots \quad \dfrac{{}^mM_l}{\mathbb{T}_{il}}} \quad \cdots \quad \dfrac{{}^0L_k}{\mathbb{T}_k}}$$

Побудуємо нове закрите табло $\mathbb{T}^*$, що починається з диз'юнкта $M_1 \vee \cdots \vee M_l$. Оскільки C є або диз'юнктом $M_1 \vee \cdots \vee M_l$ або входить в деяке $\mathbb{T}_{ii'}$, глибина C в $\mathbb{T}^*$ стане меншою, ніж глибина C в $\mathbb{T}$, призводячи до протиріччя.

Розглянемо "ескізну" версію $\mathbb{T}^*$ (позначимо її $\mathbb{T}'$):

$$\dfrac{\mathcal{S}}{\dfrac{{}^mM_1}{\mathbb{T}_{i1}} \quad \cdots \quad \dfrac{{}^mM_j}{\dfrac{{}^0L_1}{\mathbb{T}_1} \quad \cdots \quad \perp \cdot \gamma \quad \cdots \quad \dfrac{{}^0L_k}{\mathbb{T}_k}} \quad \cdots \quad \dfrac{{}^mM_l}{\mathbb{T}_{il}}}$$

Дерево $\mathbb{T}'$ не є правильно побудованим **CT**-табло, оскільки деякі гілки з $\mathbb{T}_{i1}, \ldots, \mathbb{T}_{i(j-1)}, \mathbb{T}_{i(j+1)}, \ldots, \mathbb{T}_{il}$ можуть бути закрити в $\mathbb{T}$ через уніфікацію листа з літерою ${}^0L_i$. В $\mathbb{T}'$, таке закриття вже неможливе. Переглянемо $\mathbb{T}'$ знов:

$$\dfrac{\mathcal{S}}{\dfrac{\dfrac{{}^mM_1}{\mathbb{T}_{i1}}{\vdots}{}^nL'}{\perp \cdot ({}^0L_i \leftrightarrow {}^nL')} \quad \cdots \quad \dfrac{{}^mM_j}{\dfrac{{}^0L_1}{\mathbb{T}_1} \quad \cdots \quad \perp \cdot \gamma \quad \cdots \quad \dfrac{{}^0L_k}{\mathbb{T}_k}} \quad \cdots \quad \dfrac{{}^mM_l}{\mathbb{T}_{il}}}$$

Вираз $({}^0L_i \leftrightarrow {}^nL')$ позначає тут відповідний констрейнт. Ми можемо скоригувати цей неправильний лист (і всі інші такі листи) наступним

чином:

$$\frac{\mathcal{S}}{\begin{array}{ccccc} \begin{array}{c}\dfrac{{}^mM_1}{\mathbb{T}_{i1}}\\ \vdots\\ \dfrac{{}^nL'}{\dfrac{{}^0L_1}{\mathbb{T}_1}\quad\cdots\quad\bot\cdot({}^0L_i\leftrightarrow{}^nL')\quad\cdots\quad\dfrac{{}^0L_k}{\mathbb{T}_k}}\end{array} & \cdots & \dfrac{{}^mM_j}{\dfrac{{}^0L_1}{\mathbb{T}_1}\quad\cdots\quad\bot\cdot\gamma\quad\cdots\quad\dfrac{{}^0L_k}{\mathbb{T}_k}} & \cdots & \dfrac{{}^mM_l}{\mathbb{T}_{il}}\end{array}}$$

Тепер залишається лише "освіжити" індекси змінних в копіях піддерев, повернути індекс 0 до стартового диз'юнкта, і ми отримаємо правильно побудоване закрите **CT**-табло $\mathbb{T}^*$. Зауважимо, що підстановка, яка закриває $\mathbb{T}^*$, тривіально утворюється з підстановки для $\mathbb{T}$ копіюванням та реіндексацією. Як ми вже зазначили, глибина C в $\mathbb{T}^*$ менша, ніж в $\mathbb{T}$, отже маємо протиріччя. □

3.2.4. Зв'язок між GDT та CT

Перш за все, нам необхідно перейти від формул до диз'юнктів. Ми починаємо з множини замкнених формул Γ, де немає двох кванторів по одній змінній. Операція *наївної клаузифікації* $\mathcal{C}ls$ визначається так:

$$\begin{array}{rcl} \mathcal{C}ls(\Gamma) = \bigcup_{F\in\Gamma} \mathcal{C}ls_\varepsilon(F) & \qquad & \mathcal{C}ls_{\vec{x}}(\neg\neg F) = \mathcal{C}ls_{\vec{x}}(F) \\ \mathcal{C}ls_{\vec{x}}(F\wedge G) = \mathcal{C}ls_{\vec{x}}(F)\cup\mathcal{C}ls_{\vec{x}}(G) & & \mathcal{C}ls_{\vec{x}}(\neg(F\wedge G)) = \mathcal{C}ls_{\vec{x}}(\neg F)\vee\mathcal{C}ls_{\vec{x}}(\neg G) \\ \mathcal{C}ls_{\vec{x}}(F\vee G) = \mathcal{C}ls_{\vec{x}}(F)\vee\mathcal{C}ls_{\vec{x}}(G) & & \mathcal{C}ls_{\vec{x}}(\neg(F\vee G)) = \mathcal{C}ls_{\vec{x}}(\neg F)\cup\mathcal{C}ls_{\vec{x}}(\neg G) \\ \mathcal{C}ls_{\vec{x}}(F\supset G) = \mathcal{C}ls_{\vec{x}}(\neg F)\vee\mathcal{C}ls_{\vec{x}}(G) & & \mathcal{C}ls_{\vec{x}}(\neg(F\supset G)) = \mathcal{C}ls_{\vec{x}}(F)\cup\mathcal{C}ls_{\vec{x}}(\neg G) \\ \mathcal{C}ls_{\vec{x}}(\forall\, u\, F) = \mathcal{C}ls_{\vec{x},u}(F) & & \mathcal{C}ls_{\vec{x}}(\neg\forall\, v\, F) = \mathcal{C}ls_{\vec{x}}(\neg F[\mathbf{v}(\vec{x})/v]) \\ \mathcal{C}ls_{\vec{x}}(\exists\, v\, F) = \mathcal{C}ls_{\vec{x}}(F[\mathbf{v}(\vec{x})/v]) & & \mathcal{C}ls_{\vec{x}}(\neg\exists\, u\, F) = \mathcal{C}ls_{\vec{x},u}(\neg F) \\ \mathcal{C}ls_{\vec{x}}(P(\vec{s})) = \{\, P(\vec{s})\,\} & & \mathcal{C}ls_{\vec{x}}(\neg P(\vec{s})) = \{\,\neg P(\vec{s})\,\} \end{array}$$

де $S_1\vee S_2 = \{\, C_1\vee C_2 \,|\, C_1\in S_1, C_2\in S_2\,\}$ (S_1, S_2 є множинами диз'юнктів). Зауважимо, що в правилах для $\mathcal{C}ls_{\vec{x}}(\exists\, v\, F)$ та $\mathcal{C}ls_{\vec{x}}(\neg\forall\, v\, F)$, змінна v перетворюється на сколемівський функціональний символ $\mathbf{v}$ (або константу, якщо $\vec{x}=\varepsilon$).

Наступна теорема є очевидним наслідком теореми 3.7.

Теорема 3.8. *Нехай* $\Gamma = \Delta \cup \{G\}$ *є безконфліктною множиною замкнених формул. Припустимо, що* Δ *— сумісна. Множина* Γ *суперечлива тоді і лише тоді, коли множина диз'юнктів* $\mathcal{C}ls(\Gamma)$ *може бути спростована в* **CT**. *Крім того, якщо* Γ *суперечлива, то існує* **CT***-спростування* $\mathcal{C}ls(\Gamma)$, *що починається з диз'юнкта з* $\mathcal{C}ls_{\varepsilon}(G)$.

Правильно побудоване **GDT**-дерево (або його піддерево) називається *зведеним*, якщо всі формули в його листах є літерами. Зауважимо, що будь-яке закрите дерево є зведеним. Для довільного зведеного піддерева $\mathbb{T}$ правильно побудованого **GDT**-табло, ми визначаємо *літералізацію* $\mathbb{T}$, $\mathcal{L}it(\mathbb{T})$ як послідовність дерев:

1. Якщо кореневий вузол в $\mathbb{T}$ є початковою множиною формул Γ, тоді $\mathcal{L}it(\mathbb{T})$ складається з одного дерева:

$$\mathcal{L}it\left(\frac{\Gamma}{\mathbb{T}'}\right) \;=\; \frac{\mathcal{C}ls(\Gamma)}{\mathcal{L}it(\mathbb{T}')\theta_{\Gamma}}$$

   де сколемізуюча підстановка θ_{Γ} застосована як до формул, так і до констрейнтів в літералізації $\mathcal{L}it(\mathbb{T}')$.

2. Якщо кореневий вузол в $\mathbb{T}$ містить нелітерну формулу, то $\mathcal{L}it(\mathbb{T})$ є конкатенацією:

$$\begin{aligned}
\mathcal{L}it\left(\frac{F \diamond \tau \cdot \gamma}{\mathbb{T}_1}\right) &= \mathcal{L}it(\mathbb{T}_1 + \gamma)\\
\mathcal{L}it\left(\frac{(G * H) \diamond 0.\tau \cdot \gamma}{\mathbb{T}_1 \qquad \mathbb{T}_2}\right) &= \mathcal{L}it(\mathbb{T}_1 + \gamma),\ \ \mathcal{L}it(\mathbb{T}_2)\\
\mathcal{L}it\left(\frac{(G * H) \diamond 1.\tau \cdot \gamma}{\mathbb{T}_1 \qquad \mathbb{T}_2}\right) &= \mathcal{L}it(\mathbb{T}_1),\ \ \mathcal{L}it(\mathbb{T}_2 + \gamma)\\
\mathcal{L}it\left(\frac{\neg(G * H) \diamond 0.0.\tau \cdot \gamma}{\mathbb{T}_1 \qquad \mathbb{T}_2}\right) &= \mathcal{L}it(\mathbb{T}_1 + \gamma),\ \ \mathcal{L}it(\mathbb{T}_2)\\
\mathcal{L}it\left(\frac{\neg(G * H) \diamond 0.1.\tau \cdot \gamma}{\mathbb{T}_1 \qquad \mathbb{T}_2}\right) &= \mathcal{L}it(\mathbb{T}_1),\ \ \mathcal{L}it(\mathbb{T}_2 + \gamma)
\end{aligned}$$

   де F — нелітерна формула, $*$ — бінарна пропозиційна зв'язка, а $\mathbb{T}_i + \gamma$ отримано з $\mathbb{T}_i$ доданням констрейнту γ до констрейнту в кореневому вузлі.

3. Якщо $\mathbb{T}$ містить літеру в кореневому вузлі, $\mathcal{L}it(\mathbb{T})$ складається з одного дерева. Якщо корінь $\mathbb{T}$ не має нащадків, $\mathcal{L}it(\mathbb{T})$ також складається з одного вузла.

$$\mathcal{L}it\left(\frac{L \diamond \tau \cdot \gamma}{\mathbb{T}'}\right) \;=\; \frac{L \cdot \gamma}{\mathcal{L}it(\mathbb{T}')} \qquad\qquad \mathcal{L}it\left(L \diamond \tau \cdot \gamma\right) \;=\; (L \cdot \gamma)$$

Повернемось до прикладу на рисунку 5. Починаючи з множини формул Γ, маємо:

$$\mathcal{C}ls(\Gamma) = \{\, \neg(p \subseteq q) \vee \neg(x \in p) \vee (x \in q),\ \neg E(r) \vee \neg(y \in r),\ (\mathbf{z}(s) \in s) \vee E(s),\ E(\mathbf{e}),\ \mathbf{a} \subseteq b,\ \neg E(\mathbf{a}) \,\}$$

Літералізоване дерево виглядає наступним чином (ми записуємо констрейнти під листами, щоб залишитися в межах сторінки). Отримане виведення є правильно побудованим закритим **CT**-табло:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\mathcal{C}ls(\Gamma)}{\neg E(\mathbf{a})}}{\mathbf{z}({}^1s) \in {}^1s}}{\dfrac{\neg\, {}^2p \subseteq {}^2q}{\bot}\ (\mathbf{z}({}^1s) = {}^2x \wedge {}^1s = {}^2p)}\quad \dfrac{{}^2x \in {}^2q}{\dfrac{\neg\, E({}^4r)}{\bot}\ ({}^2x = {}^4y \wedge {}^2q = {}^4r)}}{\bot}\ ({}^2p = \mathbf{a} \wedge {}^2q = {}^3b)}\quad \dfrac{\bot}{({}^4r = \mathbf{e})}}{\bot}\ (\mathbf{a} = {}^1s)$$

Поширимо визначення наївної клаузифікації на формули з індексованими змінними:

$$\mathcal{C}ls_{\vec{x}}(\forall\, {}^k u\, F) = \mathcal{C}ls_{\vec{x}}(F) \qquad \mathcal{C}ls_{\vec{x}}(\neg\forall\, {}^k\mathbf{v}\, F) = \mathcal{C}ls_{\vec{x}}(\neg F)$$
$$\mathcal{C}ls_{\vec{x}}(\exists\, {}^k\mathbf{v}\, F) = \mathcal{C}ls_{\vec{x}}(F) \qquad \mathcal{C}ls_{\vec{x}}(\neg\exists\, {}^k u\, F) = \mathcal{C}ls_{\vec{x}}(\neg F)$$

Легко бачити, що для всіх замкнених $F \in \Gamma$, $\mathcal{C}ls_{\varepsilon}({}^kF)\theta_\Gamma = {}^k(\mathcal{C}ls_{\varepsilon}(F))$.

Лема 3.9. *Нехай Γ є безконфліктна множина замкнених формул, $\mathbb{T}'$ — зведене піддерево деякого* **GDT***-табло, побудованого з Γ, і γ' — довільний констрейнт, можливо пустий. Покладемо $\mathbb{T} = \mathbb{T}' + \gamma'$ і допустимо, що $(F \diamond \tau \cdot \gamma)$ є кореневим вузлом $\mathbb{T}$. Тоді має місце наступне:*

1. *Всі вузли в $\mathcal{L}it(\mathbb{T})$ є парами (літера · констрейнт).*

2. *Всі літери з вузлів* $\mathbb{T}$ *присутні в* $\mathcal{L}it(\mathbb{T})$*; їхнє відносне розташування (бути над, бути зліва) зберігається;* $\mathcal{L}it(\mathbb{T})$ *не містить доданих літер.*

3. *Всі констрейнти з вузлів* $\mathbb{T}$ *присутні в* $\mathcal{L}it(\mathbb{T})$*;* $\mathcal{L}it(\mathbb{T})$ *не містить доданих констрейнтів.*

4. *Нехай* $L_1, \ldots, L_n$ *будуть літерами в корені дерева* $\mathcal{L}it(\mathbb{T})$*. Диз'юнкт* $(L_1 \vee \cdots \vee L_n)$ *належить до* $\mathcal{C}ls_\varepsilon(F)$*.*

5. *Нехай* L *позначає* $F|_\tau$*, якщо* $\tau \in \Pi^+(F)$*, і позначає* $\neg(F|_\tau)$*, в противному випадку. Вузол* $(L \cdot \gamma)$ *є одним з кореневих вузлів в* $\mathcal{L}it(\mathbb{T})$*.*

Доведення. Доведемо лему по індукції по числу вузлів в $\mathbb{T}$. Розглянемо кореневий вузол $\mathbb{T}$ і проведемо розбір випадків.

Випадок: $F = G \vee H$ *та* $\tau = 0.\mu$. Оскільки $\mathbb{T}$ є зведеним, то корень $\mathbb{T}$ має нащадків. Оскільки $\mathbb{T}$ є поддеревом правильно побудованого **GDT**-табло (можливо, з доданими констрейнтами), цими вузлами є $\mathbb{T}_1$ з коренем $G \diamond \mu$ та $\mathbb{T}_2$ з коренем $H \diamond \nu$, де ν є позицією атома в H. За визначенням, $\mathcal{L}it(\mathbb{T})$ є конкатенацією $\mathcal{L}it(\mathbb{T}_1 + \gamma)$ та $\mathcal{L}it(\mathbb{T}_2)$. Дерева $\mathbb{T}_1 + \gamma$ та $\mathbb{T}_2$ зведені, вони мають меншу кількість вузлів, ніж $\mathbb{T}$, отже ми можемо застосувати гіпотезу індукції. Перевіримо тепер п'ять тверджень леми:

1. Так: $\mathcal{L}it(\mathbb{T})$ є конкатенацією $\mathcal{L}it(\mathbb{T}_1 + \gamma)$ та $\mathcal{L}it(\mathbb{T}_2)$;

2. Так: $\mathbb{T}_1 + \gamma$ містить ті ж самі літери, що й $\mathbb{T}_1$, і всі літери з $\mathcal{L}it(\mathbb{T}_1 + \gamma)$ розташовані зліва від літер з $\mathcal{L}it(\mathbb{T}_2)$;

3. Так: зокрема, $\mathcal{L}it(\mathbb{T}_1 + \gamma)$ містить констрейнт γ;

4. Нехай $L_1, \ldots, L_k$ будуть літерами в кореневих вузлах $\mathcal{L}it(\mathbb{T}_1 + \gamma)$, а $L_{k+1}, \ldots, L_n$ — в кореневих вузлах $\mathcal{L}it(\mathbb{T}_2)$. За гіпотезою індукції та розширеним визначенням $\mathcal{C}ls$, $(L_1 \vee \cdots \vee L_n) \in \mathcal{C}ls_\varepsilon(F)$.

5. Зауважимо, що $F|_\tau = G|_\mu$, а дерево $\mathbb{T}_1$ має пустий констрейнт в корені. Нехай L дорівнює $G|_\mu$, якщо $\mu \in \Pi^+(G)$, і дорівнює $\neg(G|_\mu)$, в

противному випадку. За гіпотезою індукції, $(L{\cdot}\gamma)$ є одним з кореневих вузлів $\mathcal{L}it(\mathbb{T}_1 + \gamma)$.

Інші випадки для нелітерної формули F розглядаються аналогічно.

Випадок: F є літерою. Оскільки $\mathbb{T}$ побудовано згідно з правилами **GDT**, кореневий вузол $\mathbb{T}$ має щонайбільше одного нащадка. Припустимо, що є нащадок $\mathbb{T}_1$. Як ми знаємо, $\mathcal{L}it(\mathbb{T}) = (F{\cdot}\gamma)/\mathcal{L}it(\mathbb{T}_1)$. Ми можемо застосувати гіпотезу індукції для $\mathbb{T}_1$. Очевидно, п'ять тверджень леми виконуються для $\mathcal{L}it(\mathbb{T})$. Це також справедливо, якщо $\mathbb{T}$ складається з одного вузла. □

Теорема 3.10. *Нехай Γ є безконфліктна множина замкнених формул. Для довільного зведеного* **GDT***-табло $\mathbb{T}$, побудованого з Γ, $\mathcal{L}it(\mathbb{T})$ є правильно побудованим* **CT***-табло; якщо $\mathbb{T}$ закрито, то $\mathcal{L}it(\mathbb{T})$ закрито також.*

Доведення. Доведемо першу частину теореми по індукції по кількості вузлів в $\mathbb{T}$. Припустимо, що всі вузли з літерами є листами в $\mathbb{T}$. Тоді $\mathbb{T}$ має вигляд

$$\dfrac{\Gamma}{{}^0F \diamond \tau}$$
$$\vdots$$

де $F \in \Gamma$, а τ є позицією атома в F. За лемою 3.9, $\mathcal{L}it(\mathbb{T})$ має вигляд

$$\dfrac{\mathcal{C}ls(\Gamma)}{L_1\theta_\Gamma \qquad \cdots \qquad L_n\theta_\Gamma}$$

де $(L_1 \vee \cdots \vee L_n) \in \mathcal{C}ls_\varepsilon({}^0F)$. Справді, оскільки всі літери в $\mathbb{T}$ знаходяться в листах, літералізація $\mathcal{L}it(\mathbb{T})$ має глибину 2. Зауважимо, що $\mathcal{L}it(\mathbb{T})$ не містить констрейнтів, оскільки таке дерево могло бути побудовано лише за допомогою $\alpha\beta\gamma\delta$-правил, які не породжують додаткових констрейнтів.

За правилами наївної клаузифікації для формул з індексованими змінними, існує диз'юнкт $(L'_1 \vee \cdots \vee L'_n) \in \mathcal{C}ls_\varepsilon(F)$, такий що ${}^0L'_i = L_i\theta_\Gamma$ для всіх $i \in [1, n]$. Тому $\mathcal{L}it(\mathbb{T})$ є правильно побудованим **CT**-табло, утвореним з $\mathcal{C}ls(\Gamma)$ єдиним застосуванням стартового правила.

Тепер допустимо, що в $\mathbb{T}$ є нелистові вузли з літерами. Розглянемо одий такий вузол N з максимальною глибиною в $\mathbb{T}$. Припустимо, що літера в N є запереченим атомом $\neg P(\vec{r})$ (випадок позитивної літери розглядається аналогічно). Оскільки $\mathbb{T}$ є правильно побудованим **GDT**-табло, гілка з вузлом N може бути або продовжена кроком породження нової цілі, або закрита за термінальним правилом.

Припустимо, що гілка продовжена за правилом вибору нової цілі. Тоді піддерево, що росте з N має вигляд $N/\mathbb{T}_1$. Видалимо дерево $\mathbb{T}_1$ з $\mathbb{T}$ і позначимо отримане дерево $\mathbb{T}'$. Дерево $\mathbb{T}'$ є правильно побудованим зведеним **GDT**-табло, яке має меншу кількість вузлів, ніж $\mathbb{T}$. Отже, ми можемо застосувати гіпотезу індукції. За лемою 3.9, дерево $\mathcal{L}it(\mathbb{T}')$ має лист N' з літерою $\neg P(\vec{r})\theta_\Gamma$.

Піддерево $\mathbb{T}_1$ має вигляд $({}^mF[\bot]_\tau \diamond \tau \cdot \gamma)/\mathbb{T}_2$, де $F[P(\vec{s})^+]_\tau \in \Gamma$, а γ є складений констрейнт ${}^m\vec{s} = \vec{r}$. Очевидно, всі вузли з літерами в $\mathbb{T}_1$ є листами. За лемою 3.9, літералізація $\mathcal{L}it(\mathbb{T}_1)$ є набором дерев, що містять рівно один вузол $L_1, \ldots, (\bot \cdot \gamma), \ldots, L_n$, де диз'юнкт $(L_1 \vee \cdots \vee \bot \vee \cdots \vee L_n) \in \mathcal{C}ls_\varepsilon({}^mF[\bot]_\tau)$. Припустимо, що $\bot$ є k-тою літерою в диз'юнкті. Відмітимо, що $\mathcal{L}it(\mathbb{T}_1)$ не містить інших констрейнтів, крім γ, оскільки $\mathbb{T}_2$ було утворено за допомогою $\alpha\beta\gamma\delta$-правил.

Легко побачити, що диз'юнкт $(L_1 \vee \cdots \vee {}^mP(\vec{s}) \vee \cdots \vee L_n)$ належить до $\mathcal{C}ls_\varepsilon({}^mF[P(\vec{s})]_\tau)$. Тоді, за правилами наївної клаузифікації для формул з індексованими змінними, існує диз'юнкт $(L'_1 \vee \cdots \vee L'_n) \in \mathcal{C}ls_\varepsilon(F[P(\vec{s})]_\tau)$, такий що ${}^mL'_i = L_i\theta_\Gamma$ для всіх $i \in [1,n]\backslash\{k\}$ та ${}^mL'_k = ({}^mP(\vec{s}))\theta_\Gamma$. В численні **CT**, ми можемо розширити вузол N' в дереві $\mathcal{L}it(\mathbb{T}')$ вузлами ${}^mL'_1, \ldots, (\bot \cdot \gamma\theta_\Gamma), \ldots, {}^mL'_n$. За попереднім, отримане дерево буде в точності $\mathcal{L}it(\mathbb{T})$.

Припустимо теперь, що вузол N в $\mathbb{T}$ був продовжений термінальним кроком. Тоді існує вузол M в гілці N, який містить літеру $P(\vec{s})$, і піддерево, що росте з N, має вигляд $N/(\bot \diamond \varepsilon \cdot \gamma)$, де γ є складений констрейнт $\vec{s} = \vec{r}$.

Як і раніше, видалимо цей лист з $\mathbb{T}$ і позначимо отримане дерево $\mathbb{T}'$. Дерево $\mathbb{T}'$ є правильно побудоване зведене **GDT**-табло, кількість вузлів

в якому менша, ніж в $\mathbb{T}$. Отже ми можемо застосувати гіпотезу індукції. За лемою 3.9, в дереві $\mathcal{L}it(\mathbb{T}')$ є лист N' з літерою $\neg P(\vec{r})\theta_\Gamma$ і вузол M' в гілці N', який містить літеру $P(\vec{s})\theta_\Gamma$. В численні **CT**, ми можемо продовжити гілку N' в дереві $\mathcal{L}it(\mathbb{T}')$ вузлом $(\bot \cdot \gamma\theta_\Gamma)$. Отримане дерево буде в точності $\mathcal{L}it(\mathbb{T})$.

Доведемо другу частину твердження теореми. Припустимо, що $\mathbb{T}$ є закритим **GDT**-табло. Тоді кожна гілка в ньому містить вузол з $\bot$, і деяка допустима підстановка σ задовольняє сукупному констрейнту $\bar{\gamma}$ в $\mathbb{T}$. За лемою 3.9, кожна гілка в $\mathcal{L}it(\mathbb{T})$ також містить вузол з $\bot$. Більш того, сукупна множина констрейнтів в $\mathcal{L}it(\mathbb{T})$ співпадає з $\bar{\gamma}\theta_\Gamma$. За теоремою 3.3, ця множина виконувана, отже $\mathcal{L}it(\mathbb{T})$ є закритим **CT**-табло. □

Лема 3.11. *Нехай F є формулою з індексованими змінними. Нехай $(L_1 \vee \cdots \vee L_n)$ належить до $\mathcal{C}ls_\varepsilon(F)$. Для всіх $i \in [1, n]$ знайдеться така позиція атома τ в $\Pi^+(F)$ або в $\Pi^-(F)$, що $L_i = F|_\tau$ або, відповідно, $L_i = \neg(F|_\tau)$, і для всіх атомів A і правильно побудованих* **GDT***-табло $\mathbb{T}$ з листом N вигляду $(F[A]_\tau \diamond \tau \cdot \gamma)$ існує правильно побудоване* **GDT***-табло $\mathbb{T}'$, таке що:*

1. *$\mathbb{T}'$ отримано з $\mathbb{T}$ $\alpha\beta\gamma\delta$-кроками, застосованими під вузлом N.*
2. *піддерево, що росте з N в $\mathbb{T}'$, містить в листах літери $L_1, \ldots, L_{k-1}$, $L, L_{k+1}, \ldots, L_n$, де $L = A$, якщо $\tau \in \Pi^+(F)$, і $L = \neg A$, в противному випадку.*

Доведення. Цю лему легко довести по індукції по довжині формули F.

Випадок: F є літерою. Твердження очевидне: τ є або ε, або 0 (залежно від того, чи є F незапереченим або запереченим атомом) і $\mathbb{T}' = \mathbb{T}$.

Випадок: $F = G \vee H$. Для деякого $k \in [1, n]$, диз'юнкт $(L_1 \vee \cdots \vee L_k)$ належить до $\mathcal{C}ls_\varepsilon(G)$, і диз'юнкт $(L_{k+1} \vee \cdots \vee L_n)$ належить до $\mathcal{C}ls_\varepsilon(G)$. Припустимо, що $i \leqslant k$ (випадок, коли $i > k$ розглядається аналогічно). Візьмемо деяке $j \in [k+1, n]$. За гіпотезою індукції, ми можемо вибрати такі позиції $\mu \in \Pi(G)$ і $\nu \in \Pi(H)$, що задовольняють твердження леми. Покладемо $\tau = 0.\mu$. Тепер, нехай $\mathbb{T}$ буде правильно побудованим

GDT-табло з листом вигляду $(F[A]_\tau \diamond \tau \cdot \gamma)$. Ми можемо розширити $\mathbb{T}$, застосувавши β-правило до цього листа і породивши два нових вузла $(G[A]_\mu \diamond \mu)$ та $(H \diamond \nu)$. За гіпотезою індукції, ці вузли можуть бути розвинуті $\alpha\beta\gamma\delta$-кроками до відповідного дерева $\mathbb{T}'$.

Випадок: $F = G \wedge H$. Диз'юнкт $(L_1 \vee \cdots \vee L_k)$ належить до $\mathcal{C}ls_\varepsilon(G) \cup \mathcal{C}ls_\varepsilon(H)$. Припустимо, що він прийшов з $\mathcal{C}ls_\varepsilon(G)$ (інший випадок розглядається так само). Нехай i належить до $[1, n]$. За гіпотезою індукції, ми можемо вибрати підходящу позицію $\mu \in \Pi(G)$. Покладемо $\tau = 0.\mu$. Тепер, нехай $\mathbb{T}$ буде правильно побудованим **GDT**-табло з листом вигляду $(F[A]_\tau \diamond \tau \cdot \gamma)$. Ми можемо розширити $\mathbb{T}$, застосувавши α-правило до цього листа і породивши новий лист $(G[A]_\mu \diamond \mu)$. За гіпотезою індукції, цей вузол може бути розвинений $\alpha\beta\gamma\delta$-кроками до дерева $\mathbb{T}'$.

Решта випадків розглядається аналогічно. □

Теорема 3.12. *Нехай* Γ *є безконфліктна множина замкнених формул. Для довільного правильно побудованого* **CT***-табло* $\mathbb{T}^\circ$*, що побудовано з* $\mathcal{C}ls(\Gamma)$ *і має стартовий диз'юнкт з останньої формули в* Γ*, існує зведене* **GDT***-табло* $\mathbb{T}$*, таке що* $\mathbb{T}^\circ = \mathcal{L}it(\mathbb{T})$*. Якщо* $\mathbb{T}^\circ$ *закрите, то* $\mathbb{T}$ *закрите також.*

Доведення. Щоб довести першу частину твердження, проведемо індукцію по числу вузлів в $\mathbb{T}^\circ$. Припустимо, що $\mathbb{T}^\circ$ отримано з **CT**-табло $\mathbb{T}^*$ кроком розширення (випадки стартового чи термінального кроків розглядаються аналогічно) Припустимо, що $\mathbb{T}^*$ має лист N вигляду $\neg P(\vec{r})$ і існує диз'юнкт $(L_1 \vee \cdots \vee L_n) \in \mathcal{C}ls(\Gamma)$, такий що $L_k = P(\vec{s})$, і N має нащадків ${}^mL_1, \ldots, (\bot \cdot \gamma), \ldots, {}^mL_n$ в $\mathbb{T}^\circ$, де γ є складеним констрейнтом ${}^m\vec{s} = \vec{r}$.

Дерево $\mathbb{T}^*$ є правильно побудованим **CT**-табло, яке має менше вузлів, ніж $\mathbb{T}^\circ$. Тому ми можемо скористатися гіпотезою індукції. Нехай $\mathbb{T}'$ буде відповідним **GDT**-табло. Оскільки $\mathcal{L}it(\mathbb{T}') = \mathbb{T}^*$, знайдеться лист N' в $\mathbb{T}'$, що містить літеру $P(\vec{r}')$, таку що $\vec{r}'\theta_\Gamma = \vec{r}$. Також, існує формула $F \in \Gamma$ та диз'юнкт $(L'_1 \vee \cdots \vee L'_n) \in \mathcal{C}ls_\varepsilon({}^mF)$, такі що $L'_i\theta_\Gamma = {}^mL_i$ для всіх $i \in [1, n]$. Нехай $L'_k = P(\vec{s}')$.

Для заданої формули ${}^mF$, диз'юнкта $(L'_1 \vee \cdots \vee L'_n)$, та індекса k, лема 3.11 дає нам деяку позицію $\tau \in \Pi^+({}^mF)$, таку що ${}^mF|_\tau = P(\vec{s}')$. Тоді ми можемо продовжити вузол N' в **GDT**-табло $\mathbb{T}'$ нащадком $({}^mF[\bot]_\tau \diamond \tau{\cdot}\gamma')$, де γ' є складеним констрейнтом ${}^m\vec{s}' = \vec{r}'$. Зауважимо, що $\gamma'\theta_\Gamma = \gamma$. За лемою 3.11, отримане дерево може бути розвинуто $\alpha\beta\gamma\delta$-кроками під вузлом N' до листів $L'_1, \ldots, \bot, \ldots, L'_n$ (якщо F сама є літерою, додаткові кроки не потрібні). Позначимо через $\mathbb{T}$ отримане **GDT**-tree. Легко побачити, що $\mathcal{L}it(\mathbb{T})$ може бути отримане з $\mathcal{L}it(\mathbb{T}')$ розширенням листа N вузлами $L'_1\theta_\Gamma, \ldots, (\bot \cdot \gamma'\theta_\Gamma), \ldots L'_n\theta_\Gamma$. Отже, $\mathcal{L}it(\mathbb{T}) = \mathbb{T}^\circ$.

Друга частина теореми випливає з леми 3.9 та теореми 3.4. □

Загалом, теореми 3.10, 3.12, 3.8 доводять коректність і повноту числення **GDT** (теорема 3.5).

3.3. Обробка рівності

В цьому розділі розвивається коректне і повне цілекероване табличне числення для класичної логіки першого порядку з рівністю. Для спрощення викладення, ми працюватимо з диз'юнктами: досліджуване числення **LPCT** є, насправді, модифікацією числення **CT**, яке розширено правилами виведення для обробкі рівності. Попередній розділ демонструє, яким чином переформулювати запропоноване числення в термінах формул першого порядку та допустимих підстановок.

Позначатимемо рівність символом $\approx$. Будемо ототожнювати атоми рівності, що розрізняються лише порядком аргументів. Наприклад, атом $f(x) \approx y$ уніфікуватиметься з $0 \approx f(1)$. Символ $\simeq$ позначатиме "псевдорівність", бінарний предикат, що не має спеціального логічного змісту. Цей символ використовується для того, щоб замінити символ $\approx$, коли ми переходимо від логіки з рівністю до логіки без рівності. На відміну від рівності, $f(x) \simeq y$ не уніфікується з $0 \simeq f(1)$.

В подальшому ми позначаємо терми, які не є змінними, літерами p та q, а довільні терми літерами l, r, s, та t.

3.3.1. Парамодуляція і цілекерованість

На рисунку 7 викладено числення $\mathbf{CT}^{\approx}$, яке є простим розширенням числення $\mathbf{CT}$ правилом парамодуляції, яке застосовується в цілекерованій манері. Нагадаємо, що змінні, які входять в початкові диз'юнкти не є індексованими, а змінні, які входять в породжені вузли табло, обов'язково індексовані.

На жаль, числення $\mathbf{CT}^{\approx}$ неповне. Розглянемо таку несумісну множину диз'юнктів:

$$\mathcal{S} = \{a \approx b,\ \ c \approx d,\ \ \neg P(f(a), f(b)),\ \ \neg Q(g(c), g(d)),\ \ P(x,x) \vee Q(y,y)\}$$

Спробуємо побудувати спростування $\mathcal{S}$ в $\mathbf{CT}^{\approx}$:

$$\dfrac{\dfrac{\dfrac{\dfrac{\mathcal{S}}{a \approx b}\,(\mathrm{St})}{\neg P(f(b), f(b)) \cdot (a = a)}\,(\mathrm{EqEx.2})}{\bot \cdot ({}^1x = f(b)) \qquad \dfrac{Q({}^1y, {}^1y)}{?}}\,(\mathrm{Ex.2}) \qquad \dfrac{\dfrac{\dfrac{\dfrac{\mathcal{S}}{\neg Q(g(c), g(d))}\,(\mathrm{St})}{\neg Q(g(d), g(d)) \cdot (c = c)}\,(\mathrm{EqEx.1})}{\dfrac{P({}^1x, {}^1x)}{?} \qquad \bot \cdot ({}^1y = g(d))}\,(\mathrm{Ex.2})}$$

Ми не можемо продовжити перше виведення, оскільки літера $Q({}^1y, {}^1y)$ не може бути уніфікована з $Q(g(c), g(d))$; так само і рівність $c = d$ не може бути застосована до неї. Друге виведення обривається схожим чином.

Проблема полягає в тому, що ліфтінг з основних термів на терми зі змінними неможливий в виведеннях з парамодуляцією. Наш контрприклад не спрацював би, якби змінна x (або y) була інстанційована. Звісно, повноти можна було б досягнути, додавши аксіоми *функціональної рефлексивності* ($f(x) \approx f(x)$, $g(x) \approx g(x)$, і так далі) і дозволити парамодуляцію в змінні. Втім, такий підхід виявляється досить неефективним на практиці, оскільки функціональна рефлексивність може бути застосована в будь-який момент виведення.

Числення резолюційного типу сумісні з правилом парамодуляції завдяки своїй гнучкості щодо порядку кроків виведення, яка є недосяжною в цілекерованих численнях. Огляд існуючих підходів до обробки рівності в численнях секвенціального та табличного типу проведено в [56].

Правило старту:

$$\frac{\Gamma,\ (L_1 \vee \cdots \vee L_k),\ \Lambda \parallel}{{}^0L_1 \quad \cdots \quad {}^0L_k}$$

Правило редукції:

$$\frac{\Gamma \parallel \Delta, s \not\approx t}{\bot \cdot (s = t)}$$

Правила розширення (m є новим індексом):

$$\frac{\Gamma,\ (L_1 \vee \cdots \vee \neg P(\vec{s}) \vee \cdots \vee L_k),\ \Lambda \parallel \Delta, P(\vec{r})}{{}^mL_1 \quad \cdots \quad \bot \cdot ({}^ms_1 = r_1 \wedge \cdots \wedge {}^ms_n = r_n) \quad \cdots \quad {}^mL_k}$$

$$\frac{\Gamma,\ (L_1 \vee \cdots \vee P(\vec{s}) \vee \cdots \vee L_k),\ \Lambda \parallel \Delta, \neg P(\vec{r})}{{}^mL_1 \quad \cdots \quad \bot \cdot ({}^ms_1 = r_1 \wedge \cdots \wedge {}^ms_n = r_n) \quad \cdots \quad {}^mL_k}$$

Розширення з рівністю (m є новим індексом):

$$\frac{\Gamma,\ (L_1 \vee \cdots \vee l \approx r \vee \cdots \vee L_k),\ \Lambda \parallel \Delta, L[p]}{{}^mL_1 \quad \cdots \quad L[{}^mr] \cdot (p = {}^ml) \quad \cdots \quad {}^mL_k}$$

$$\frac{\Gamma,\ (L_1 \vee \cdots \vee L[p] \vee \cdots \vee L_k),\ \Lambda \parallel \Delta, l \approx r}{{}^mL_1 \quad \cdots \quad {}^mL[r] \cdot ({}^mp = l) \quad \cdots \quad {}^mL_k}$$

Правила парамодуляції:

$$\frac{\Gamma \parallel \Delta, L[p], \Lambda, l \approx r}{L[r] \cdot (p = l)} \qquad \frac{\Gamma \parallel \Delta, l \approx r, \Lambda, L[p]}{L[r] \cdot (p = l)}$$

Термінальні правила:

$$\frac{\Gamma \parallel \Delta, \neg P(\vec{s}), \Lambda, P(\vec{r})}{\bot \cdot (s_1 = r_1 \wedge \cdots \wedge s_n = r_n)} \qquad \frac{\Gamma \parallel \Delta, P(\vec{s}), \Lambda, \neg P(\vec{r})}{\bot \cdot (s_1 = r_1 \wedge \cdots \wedge s_n = r_n)}$$

Рис. 7: Цілекероване числення з парамодуляцією $\mathbf{CT}^{\approx}$

3.3.2. Правило лінивої парамодуляції

Ми сказали вище, що літера $Q({}^1y, {}^1y)$ не може бути уніфікована з літерою $Q(g(c), g(d))$. Але що, як ми відкладемо уніфікацію до того, як рівність $c \approx d$ буде застосована до другої літери? Що станеться, якщо зробити рівність $Q({}^1y, {}^1y) = Q(g(c), g(d))$ не констрейнтом, що має бути задоволений, а підцілью, яка має бути доведена? Саме це робить правило *лінивої парамодуляції*.

На рисунку 8 ми розглядаємо ескізну версію цілекерованого табличного числення з правилом лінивої парамодуляції. Приклад з попереднього підрозділу легко може бути спростований в такому численні:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\mathcal{S}}{a \approx b}}{\neg P(f(b), f(b))}}{\bot}\quad \dfrac{\dfrac{{}^1x \not\approx f(b)}{\bot \cdot ({}^1x = f(b))}}{Q({}^1y, {}^1y)}\quad \dfrac{\dfrac{\dfrac{g(c) \not\approx {}^1y}{\dfrac{g(d) \not\approx {}^1y}{\bot \cdot (g(d) = {}^1y)}}\quad \dfrac{c \not\approx c}{\bot \cdot (c = c)}}{\bot}\quad \dfrac{g(d) \not\approx {}^1y}{\bot \cdot (g(d) = {}^1y)}}{\bot}\ (\text{Ex.1})}\quad \dfrac{a \not\approx a}{\bot \cdot (a = a)}$$

Хоча обраний підхід здається придатним, запропоноване числення не більш ефективне, ніж пряме використання функціональної рефлексивності. Справді, за наведеними правилами виведення, ми можемо застосувати будь-яку рівність до будь-якого терма, який не є змінною. Чи можна вдосконалити метод? Чи є він принаймні повним?

3.3.3. Елімінація рівності з констрейнтами

Техніка CEE (Constrained Equality Elimination) є варіантом методу модифікації Бранда [57]. Вона дозволяє трансформувати множину диз'юнктів з логічним символом рівності в еквісумісну множину без такого символа, тобто перевести проблему в класичній логіці першого порядку з рівністю в класичну логіку першого порядку без рівності. Цей метод застосовує групу переписуючих правил до диз'юнктів в початковій множині $\mathcal{S}$. В результаті, ми отримуємо множину CEE($\mathcal{S}$) пласких диз'юнктів з констрейнтами, де знак рівності $\approx$ замінено на звичайний предикатний символ $\simeq$.

Правило старту:

$$\frac{\Gamma,\ (L_1 \vee \cdots \vee L_k),\ \Lambda \parallel}{{}^0L_1 \quad \cdots \quad {}^0L_k}$$

Правило редукції:

$$\frac{\Gamma \parallel \Delta, s \not\approx t}{\bot \cdot (s = t)}$$

Правила розширення (m є новим індексом):

$$\frac{\Gamma,\ (L_1 \vee \cdots \vee \neg P(\vec{s}) \vee \cdots \vee L_k),\ \Lambda \parallel \Delta, P(\vec{r})}{{}^mL_1 \quad \cdots \quad \bot \quad {}^ms_1 \not\approx r_1 \quad \cdots \quad {}^ms_n \not\approx r_n \quad \cdots \quad {}^mL_k}$$

$$\frac{\Gamma,\ (L_1 \vee \cdots \vee P(\vec{s}) \vee \cdots \vee L_k),\ \Lambda \parallel \Delta, \neg P(\vec{r})}{{}^mL_1 \quad \cdots \quad \bot \quad {}^ms_1 \not\approx r_1 \quad \cdots \quad {}^ms_n \not\approx r_n \quad \cdots \quad {}^mL_k}$$

Розширення з рівністю (m є новим індексом):

$$\frac{\Gamma,\ (L_1 \vee \cdots \vee l \approx r \vee \cdots \vee L_k),\ \Lambda \parallel \Delta, L[p]}{{}^mL_1 \quad \cdots \quad L[{}^mr] \quad p \not\approx {}^ml \quad \cdots \quad {}^mL_k}$$

$$\frac{\Gamma,\ (L_1 \vee \cdots \vee L[p] \vee \cdots \vee L_k),\ \Lambda \parallel \Delta, l \approx r}{{}^mL_1 \quad \cdots \quad {}^mL[r] \quad {}^mp \not\approx l \quad \cdots \quad {}^mL_k}$$

Правила парамодуляції:

$$\frac{\Gamma \parallel \Delta, L[p], \Lambda, l \approx r}{L[r] \qquad p \not\approx l} \qquad \frac{\Gamma \parallel \Delta, l \approx r, \Lambda, L[p]}{L[r] \qquad p \not\approx l}$$

Термінальні правила:

$$\frac{\Gamma \parallel \Delta, \neg P(\vec{s}), \Lambda, P(\vec{r})}{\bot \quad s_1 \not\approx r_1 \quad \cdots \quad s_n \not\approx r_n} \qquad \frac{\Gamma \parallel \Delta, P(\vec{s}), \Lambda, \neg P(\vec{r})}{\bot \quad s_1 \not\approx r_1 \quad \cdots \quad s_n \not\approx r_n}$$

Рис. 8: Цілекероване числення з лінивою парамодуляцією (ескіз)

В цій роботі ми викладаємо правила CEE в дещо зміненій формі, порівняно з оригінальною роботою Бахмайра та ін. [58]. По-перше, ми дозволяємо інші предикатні символи, крім рівності. По-друге, ми застосовуємо елімінацію монотонності після елімінації симетрії. По-третє, ми працюємо з традиційними диз'юнктами і переносимо констрейнти до правил виведення. По-четверте, ми обробляємо входження запереченої рівності змінних $x \not\approx y$ в дещо інший спосіб. Втім, усі ці модифікації носять суто технічний характер і не впливають на головний результат (теорема 3.13).

Чотири групи переписуючих правил на рисунку 9 застосовуються в порядку викладення. Змінні з кришечкою вважаються новими в відповідному диз'юнкті. Нагадаємо, що литери p та q позначають терми, які не є змінними, тоді як літери l, r, s, та t позначають довільні терми.

Перша група переписуючих правил замінює знак рівності $\approx$ на предикатний символ $\simeq$, та позбавляє нас потреби в явних аксіомах симетричності для символу $\simeq$. Друга група сплощує терми і, таким чином, відкидає необхідність в явних аксіомах монотонності для $\simeq$. Третя група розбиває літери рівності, позбавляючи нас потреби в явних аксиомах транзитивності. Останнє правило явно додає аксіому рефлексивності до множини диз'юнктів.

В отриманій множині диз'юнктів CEE($\mathcal{S}$), резолюції відповідають парамодуляціям в початковій множині. Додаткові змінні є, в якомусь сенсі, "значеннями" термів в лівій частині нових нерівностей. Під "значенням" ми розуміємо тут результат всіх парамодуляцій, зроблених в терм і всередину терму.

Зафіксуємо деяке повне фундоване впорядкування $\prec$ на основних термах. Будемо називати *основним констрейнтним прикладом* аксіоми рефлексивності $z \simeq z$ довільний основний диз'юнкт виду $p \simeq p$. Будемо називати *основним констрейнтним прикладом* будь-якого іншого диз'юнкта C довільний основний диз'юнкт $C\sigma$, такий, що для кожної позитивної літери $s \simeq t$ в C, $s\sigma \succ t\sigma$, для кожної негативної літери $s \not\simeq t$ в C, $s\sigma \succeq t\sigma$, і для кожної негативної літери $x \not\simeq y$ в C, $x\sigma = y\sigma$. Назве-

1. Елімінація симетрії:

$$\frac{L_1 \vee \cdots \vee s \approx t \vee \cdots \vee L_n}{L_1 \vee \cdots \vee s \simeq t \vee \cdots \vee L_n} \qquad \frac{L_1 \vee \cdots \vee s \approx t \vee \cdots \vee L_n}{L_1 \vee \cdots \vee t \simeq s \vee \cdots \vee L_n}$$

$$\frac{L_1 \vee \cdots \vee p \not\approx s \vee \cdots \vee L_n}{L_1 \vee \cdots \vee p \not\simeq s \vee \cdots \vee L_n} \qquad \frac{L_1 \vee \cdots \vee x \not\approx q \vee \cdots \vee L_n}{L_1 \vee \cdots \vee q \not\simeq x \vee \cdots \vee L_n}$$

2. Елімінація монотонності:

$$\frac{L_1 \vee \cdots \vee P(s_1, \ldots, p, \ldots, s_n) \vee \cdots \vee L_n}{L_1 \vee \cdots \vee P(s_1, \ldots, \hat{u}, \ldots, s_n) \vee p \not\simeq \hat{u} \vee \cdots \vee L_n}$$

$$\frac{L_1 \vee \cdots \vee \neg P(s_1, \ldots, p, \ldots, s_n) \vee \cdots \vee L_n}{L_1 \vee \cdots \vee \neg P(s_1, \ldots, \hat{u}, \ldots, s_n) \vee p \not\simeq \hat{u} \vee \cdots \vee L_n}$$

$$\frac{L_1 \vee \cdots \vee f(s_1, \ldots, p, \ldots, s_n) \simeq t \vee \cdots \vee L_n}{L_1 \vee \cdots \vee f(s_1, \ldots, \hat{u}, \ldots, s_n) \simeq t \vee p \not\simeq \hat{u} \vee \cdots \vee L_n}$$

$$\frac{L_1 \vee \cdots \vee f(s_1, \ldots, p, \ldots, s_n) \not\simeq t \vee \cdots \vee L_n}{L_1 \vee \cdots \vee f(s_1, \ldots, \hat{u}, \ldots, s_n) \not\simeq t \vee p \not\simeq \hat{u} \vee \cdots \vee L_n}$$

$$\frac{L_1 \vee \cdots \vee t \simeq f(s_1, \ldots, p, \ldots, s_n) \vee \cdots \vee L_n}{L_1 \vee \cdots \vee t \simeq f(s_1, \ldots, \hat{u}, \ldots, s_n) \vee p \not\simeq \hat{u} \vee \cdots \vee L_n}$$

$$\frac{L_1 \vee \cdots \vee t \not\simeq f(s_1, \ldots, p, \ldots, s_n) \vee \cdots \vee L_n}{L_1 \vee \cdots \vee t \not\simeq f(s_1, \ldots, \hat{u}, \ldots, s_n) \vee p \not\simeq \hat{u} \vee \cdots \vee L_n}$$

3. Елімінація транзитивності:

$$\frac{L_1 \vee \cdots \vee t \simeq q \vee \cdots \vee L_n}{L_1 \vee \cdots \vee t \simeq \hat{u} \vee q \not\simeq \hat{u} \vee \cdots \vee L_n}$$

$$\frac{L_1 \vee \cdots \vee p \not\simeq q \vee \cdots \vee L_n}{L_1 \vee \cdots \vee p \not\simeq \hat{u} \vee q \not\simeq \hat{u} \vee \cdots \vee L_n}$$

4. Введення рефлексивності:

$$\frac{}{z \simeq z}$$

Рис. 9: Елімінація рівності з констрейнтами

мо множину диз'юнктів *CEE-сумісною*, якщо множина всіх її основних констрейнтних прикладів є сумісною.

Наступне твердження є прямим наслідком теореми 4.1 з [58].

Факт 3.13. *Множина диз'юнктів $\mathcal{S}$ є сумісною в класичній логіці першого порядку з рівністю тоді і тільки тоді, коли множина* $\mathrm{CEE}(\mathcal{S})$ *є CEE-сумісною в класичній логіці першого порядку без рівності.*

Числення $\mathbf{CT}^{\simeq}$ на рисунку 10 є численням $\mathbf{CT}$, до якого додано констрейнти впорядкування для літер рівності. Ці констрейнти інтерпретуються відносно впорядкування $\prec$.

Теорема 3.14. *Множина диз'юнктів $\mathcal{S}$ є суперечливою в класичній логіці першого порядку з рівністю тоді і лише тоді, коли множина* $\mathrm{CEE}(\mathcal{S})$ *може бути спростована в численні* $\mathbf{CT}^{\simeq}$.

Доведення. Дамо лише загальний нарис доведення, оскільки деталі досить очевидні. По-перше, покажемо коректність $\mathbf{CT}^{\simeq}$ по відношенню до CEE-перетворених множин диз'юнктів. Розглянемо закрите $\mathbf{CT}^{\simeq}$-дерево $\mathbb{T}$, що спростовує множину $\mathrm{CEE}(\mathcal{S})$.

Підстановка, яка задовольнить сукупну множину констрейнтів $\mathbb{T}$, може бути доповнена до основної підстановки. Таким чином, ми отримаємо множину основних прикладів диз'юнктів з $\mathrm{CEE}(\mathcal{S})$. Очевидно, ця множина є суперечливою в класичній логіці першого порядку без рівності, оскільки ми можемо перетворити $\mathbb{T}$ на закрите $\mathbf{CT}$-спростування, просто викресливши констрейнти.

Легко побачити, що ці основні приклади є коректними основними констрейнтними прикладами, про які йдеться в теоремі 3.13. Справді, кожна позитивна літера $(l \simeq r)$ (окрім аксіоми рефлексивності), що бере участь в виведенні, отримує відповідний констрейнт строгої нерівності $(l > r)$ під час кроку розширення або термінального кроку. Далі, кожна негативна літера $(s \not\simeq t)$ або закривається кроком редукції (резолюція з аксіомою рефлексивності), або підпадає під резолюцію з деякою позитивною літерою рівності. В обох випадках констрейнт $(s \geqslant t)$ буде задо-

В правилах старту та розширення обраний диз'юнкт не є $(z \simeq z)$

Правило старту:

$$\frac{\Gamma,\ (L_1 \vee \cdots \vee L_k),\ \Lambda \parallel}{{}^0L_1 \quad \cdots \quad {}^0L_k}$$

Правило редукції:

$$\frac{\Gamma,\ (z \simeq z),\ \Lambda \parallel \Delta, s \not\simeq t}{\bot \cdot (s = t)}$$

Правила розширення (m є новим індексом):

$$\frac{\Gamma,\ (L_1 \vee \cdots \vee \neg P(\vec{s}) \vee \cdots \vee L_k),\ \Lambda \parallel \Delta, P(\vec{r})}{{}^mL_1 \quad \cdots \quad \bot \cdot ({}^ms_1 = r_1 \wedge \cdots \wedge {}^ms_n = r_n) \quad \cdots \quad {}^mL_k}$$

$$\frac{\Gamma,\ (L_1 \vee \cdots \vee P(\vec{s}) \vee \cdots \vee L_k),\ \Lambda \parallel \Delta, \neg P(\vec{r})}{{}^mL_1 \quad \cdots \quad \bot \cdot ({}^ms_1 = r_1 \wedge \cdots \wedge {}^ms_n = r_n) \quad \cdots \quad {}^mL_k}$$

$$\frac{\Gamma,\ (L_1 \vee \cdots \vee p \not\simeq t \vee \cdots \vee L_k),\ \Lambda \parallel \Delta, l \simeq r}{{}^mL_1 \quad \cdots \quad \bot \cdot ({}^mp = l \wedge l > r \wedge r = {}^mt) \quad \cdots \quad {}^mL_k}$$

$$\frac{\Gamma,\ (L_1 \vee \cdots \vee l \simeq r \vee \cdots \vee L_k),\ \Lambda \parallel \Delta, p \not\simeq t}{{}^mL_1 \quad \cdots \quad \bot \cdot (p = {}^ml \wedge {}^ml > {}^mr \wedge {}^mr = t) \quad \cdots \quad {}^mL_k}$$

Термінальні правила:

$$\frac{\Gamma \parallel \Delta, \neg P(\vec{s}), \Lambda, P(\vec{r})}{\bot \cdot (s_1 = r_1 \wedge \cdots \wedge s_n = r_n)} \qquad \frac{\Gamma \parallel \Delta, P(\vec{s}), \Lambda, \neg P(\vec{r})}{\bot \cdot (s_1 = r_1 \wedge \cdots \wedge s_n = r_n)}$$

$$\frac{\Gamma \parallel \Delta, p \not\simeq t, \Lambda, l \simeq r}{\bot \cdot (p = l \wedge l > r \wedge r = t)} \qquad \frac{\Gamma \parallel \Delta, l \simeq r, \Lambda, p \not\simeq t}{\bot \cdot (p = l \wedge l > r \wedge r = t)}$$

Рис. 10: Диз'юнктне цілекероване числення з констрейнтами $\mathbf{CT}^{\simeq}$

вільнено. Нарешті, нерівність $(x \not\simeq y)$ може бути лише закрита кроком редукції, отже констрейнт $(x = y)$ буде також виконано.

Тепер доведемо повноту $\mathbf{CT}^{\simeq}$ по відношенню до CEE-перетворених множин диз'юнктів. Розглянемо множину S основних констрейнтних прикладів диз'юнктів з CEE($\mathcal{S}$). За теоремою 3.13, S є суперечливою в класичній логіці першого порядку без рівності. Отже, ми можемо побудувати закрите **CT**-табло, що спростовує S, і таке, що аксіома рефлексивності $(z \simeq z)$ не є в ньому стартовим диз'юнктом (теорема 3.7). Після цього ми лише піднімаємо виведення до першого порядку і перетворюємо **CT**-табло на закрите $\mathbf{CT}^{\simeq}$-спростування CEE($\mathcal{S}$). □

3.3.4. Цілекероване числення з лінивою парамодуляцією

Тепер ми можемо запропонувати вдосконалену версію числення, що було намічене в підрозділі 3.3.2. Правила виведення числення **LPCT** (Lazy Paramodulation Connection Tableau) викладені на рисунку 11. Змінні з надкресленням вважаються новими по відношенню до всього дерева виведення і не індексуються. Констрейнт виду $(a = b > c)$ є скороченим записом констрейнту $(a = b \wedge b > c)$.

Запропоноване числення є більш складним, але також більш ефективним, ніж попередня версія. Перш за все, ми застосовуємо лініву парамодуляцію лише у кроках розширення. Правила парамодуляції та термінальні правила не відкладають уніфікацію. По-друге, сама "лінивість" тепер більш обмежена: будь-які два терма, що їхню уніфікацію ми відкладаємо, повинні мати загальний головний символ (втім, ця форма уніфікації все ще є досить слабкою, навіть порівняно з іншим способом лінивої уніфікації, запропонованим Лінчем та Снайдером [59]). По-третє, ми додаємо констрейнти впорядкування.

Коректність **LPCT** може бути показана безпосередньо, перевіркою того, що кожне правило виведення породжує лише логічні наслідки початкової множини диз'юнктів і поточної гілки.

Доведемо повноту **LPCT** перетворенням $\mathbf{CT}^{\simeq}$-спростування множини CEE-перетворених диз'юнктів у **LPCT**-спростування початкової мно-

Правило старту:

$$\frac{\Gamma,\ (L_1 \vee \cdots \vee L_k),\ \Lambda \parallel}{{}^0L_1 \quad \cdots \quad {}^0L_k}$$

Правило редукції:

$$\frac{\Gamma \parallel \Delta, s \not\approx t}{\bot \cdot (s = t)}$$

Правила розширення (m є новим індексом):

$$\frac{\Gamma,\ (L_1 \vee \cdots \vee \neg P(\vec{s}) \vee \cdots \vee L_k),\ \Lambda \parallel \Delta, P(\vec{r})}{{}^mL_1 \quad \cdots \quad \bot \cdot (\bar{v}_1 = r_1 \wedge \cdots \wedge \bar{v}_n = r_n) \quad {}^ms_1 \not\approx \bar{v}_1 \quad \cdots \quad {}^ms_n \not\approx \bar{v}_n \quad \cdots \quad {}^mL_k}$$

$$\frac{\Gamma,\ (L_1 \vee \cdots \vee P(\vec{s}) \vee \cdots \vee L_k),\ \Lambda \parallel \Delta, \neg P(\vec{r})}{{}^mL_1 \quad \cdots \quad \bot \cdot (\bar{v}_1 = r_1 \wedge \cdots \wedge \bar{v}_n = r_n) \quad {}^ms_1 \not\approx \bar{v}_1 \quad \cdots \quad {}^ms_n \not\approx \bar{v}_n \quad \cdots \quad {}^mL_k}$$

Розширення з рівністю (m є новим індексом):

$$\frac{\Gamma,\ (L_1 \vee \cdots \vee z \approx r \vee \cdots \vee L_k),\ \Lambda \parallel \Delta, L[p]}{{}^mL_1 \quad \cdots \quad L[\bar{w}] \cdot (p = {}^mz > \bar{w}) \quad {}^mr \not\approx \bar{w} \quad \cdots \quad {}^mL_k}$$

$$\frac{\Gamma,\ (L_1 \vee \cdots \vee f(\vec{s}) \approx r \vee \cdots \vee L_k),\ \Lambda \parallel \Delta, L[p]}{{}^mL_1 \quad \cdots \quad L[\bar{w}] \cdot (p = f(\vec{\bar{v}}) > \bar{w}) \quad {}^mr \not\approx \bar{w} \quad {}^ms_1 \not\approx \bar{v}_1 \quad \cdots \quad {}^ms_n \not\approx \bar{v}_n \quad \cdots \quad {}^mL_k}$$

$$\frac{\Gamma,\ (L_1 \vee \cdots \vee L[f(\vec{s})] \vee \cdots \vee L_k),\ \Lambda \parallel \Delta, l \approx r}{{}^mL_1 \quad \cdots \quad {}^mL[\bar{w}] \cdot (f(\vec{\bar{v}}) = l > r = \bar{w}) \quad {}^ms_1 \not\approx \bar{v}_1 \quad \cdots \quad {}^ms_n \not\approx \bar{v}_n \quad \cdots \quad {}^mL_k}$$

Правила парамодуляції:

$$\frac{\Gamma \parallel \Delta, L[p], \Lambda, l \approx r}{L[\bar{w}] \cdot (p = l > r = \bar{w})} \qquad \frac{\Gamma \parallel \Delta, l \approx r, \Lambda, L[p]}{L[\bar{w}] \cdot (p = l > r = \bar{w})}$$

Термінальні правила:

$$\frac{\Gamma \parallel \Delta, \neg P(\vec{s}), \Lambda, P(\vec{r})}{\bot \cdot (s_1 = r_1 \wedge \cdots \wedge s_n = r_n)} \qquad \frac{\Gamma \parallel \Delta, P(\vec{s}), \Lambda, \neg P(\vec{r})}{\bot \cdot (s_1 = r_1 \wedge \cdots \wedge s_n = r_n)}$$

Рис. 11: Цілекероване числення з лінивою парамодуляцією **LPCT**

жини диз'юнктів.

Теорема 3.15. *Нехай множина диз'юнктів $\mathcal{S}$ є суперечливою в класичній логіці першого порядку з рівністю. Тоді існує спростування $\mathcal{S}$ в* **LPCT**.

Доведення. За теоремою 3.14, можемо побудувати закрите $\mathbf{CT}^{\simeq}$-дерево $\mathbb{T}$, що спростовує множину CEE($\mathcal{S}$). Введемо проміжне числення $\mathbf{LPCT}^{\simeq}$, чиї правила виведення є правилами **LPCT**, де знаки $\approx$ та $\not\approx$ замінені, відповідно, на $\simeq$ та $\not\simeq$.

На першому етапі, ми перетворимо $\mathbb{T}$ на закрите $\mathbf{LPCT}^{\simeq}$-табло $\mathbb{T}^{(0)}$, побудоване з початкової множини CEE($\mathcal{S}$). На рисунку 12 показано як термінальні правила та правила розширення $\mathbf{CT}^{\simeq}$ можуть бути імітовані в $\mathbf{LPCT}^{\simeq}$, так, що породжені листи в відкритих гілках і сукупна множина констрейнтів залишаються тими самими. Правила старту та редукції не відрізняються в цих двох численнях. Відмітимо, що за визначенням CEE, наступне має місце для всіх диз'юнктів з CEE($\mathcal{S}$):

- В усіх атомах, що не є рівностями, аргументи є змінними.
- В усіх термах, що не є змінними, аргументи є змінними.
- Всі атоми рівності в CEE($\mathcal{S}$) мають змінну справа від знака рівності.

На другому етапі ми відновлюємо аргументи в термах. Нехай $\hat{u}$ є змінна, що була введена протягом CEE-перетворення і входить в $\mathbb{T}^{(0)}$. Зауважимо, що кожна змінна з кришечкою входить двічі в літери свого диз'юнкта і ніколи не опиняється зліва від знака рівності. Очевидно, $\hat{u}$ потрапила до $\mathbb{T}^{(0)}$ всередині деякого диз'юнкта вигляду $(\cdots \vee L[\hat{u}] \vee \cdots \vee p \not\approx \hat{u} \vee \cdots)$. Втім, необов'язково, що обидві ці літери присутні в $\mathbb{T}^{(0)}$, оскільки одна з них могла бути головною літерою на кроці розширення або розширення з рівністю. Тим не менше, ми можемо стверджувати, що $\mathbb{T}^{(0)}$ містить два неперетинаючихся піддерева, $\mathbb{T}^{\circ}$ та $\mathbb{T}^{\bullet}$, таких, що має місце наступне:

- Обидва піддерева $\mathbb{T}^{\circ}$ та $\mathbb{T}^{\bullet}$ введені на одному кроці: старту або розширення.

$\mathbf{CT}^{\simeq}$-термінальні $\Longrightarrow$ $\mathbf{LPCT}^{\simeq}$-парамодуляція:

$$\frac{\Gamma \parallel \Delta, f(\vec{x}) \not\simeq y, \Lambda, l \simeq u}{\bot \cdot (f(\vec{x}) = l > u = y)} \quad \Longrightarrow \quad \frac{\Gamma \parallel \Delta, f(\vec{x}) \not\simeq y, \Lambda, l \simeq u}{\dfrac{\bar{w} \not\simeq y \cdot (f(\vec{x}) = l > u = \bar{w})}{\bot \cdot (\bar{w} = y)}}$$

$\mathbf{CT}^{\simeq}$-розширення $\Longrightarrow$ $\mathbf{LPCT}^{\simeq}$-розширення:

$$\frac{\Gamma,\ (L_1 \vee \cdots \vee \neg P(\vec{x}) \vee \cdots \vee L_k),\ \Lambda \parallel \Delta, P(\vec{y})}{{}^mL_1 \quad \cdots \quad \bot \cdot ({}^mx_1 = y_1 \wedge \cdots \wedge {}^mx_n = y_n) \quad \cdots \quad {}^mL_k} \quad \Longrightarrow$$

$$\frac{\Gamma,\ (L_1 \vee \cdots \vee \neg P(\vec{x}) \vee \cdots \vee L_k),\ \Lambda \parallel \Delta, P(\vec{y})}{{}^mL_1 \quad \cdots \quad \bot \cdot (\bar{v}_1 = y_1 \wedge \cdots \wedge \bar{v}_n = y_n) \quad \dfrac{{}^mx_1 \not\simeq \bar{v}_1}{\bot \cdot ({}^mx_1 = \bar{v}_1)} \quad \cdots \quad \dfrac{{}^mx_n \not\simeq \bar{v}_n}{\bot \cdot ({}^mx_n = \bar{v}_n)} \quad \cdots \quad {}^mL_k}$$

$\mathbf{CT}^{\simeq}$-розширення $\Longrightarrow$ $\mathbf{LPCT}^{\simeq}$-розширення-з-рівністю:

$$\frac{\Gamma,\ (L_1 \vee \cdots \vee f(\vec{x}) \not\simeq y \vee \cdots \vee L_k),\ \Lambda \parallel \Delta, l \simeq u}{{}^mL_1 \quad \cdots \quad \bot \cdot ({}^mf(\vec{x}) = l > u = {}^my) \quad \cdots \quad {}^mL_k} \quad \Longrightarrow$$

$$\frac{\Gamma,\ (L_1 \vee \cdots \vee f(\vec{x}) \not\simeq y \vee \cdots \vee L_k),\ \Lambda \parallel \Delta, l \simeq u}{{}^mL_1 \quad \cdots \quad \dfrac{\bar{w} \not\simeq {}^my \cdot (f(\vec{v}) = l > u = \bar{w})}{\bot \cdot (\bar{w} = {}^my)} \quad \dfrac{{}^mx_1 \not\simeq \bar{v}_1}{\bot \cdot ({}^mx_1 = \bar{v}_1)} \quad \cdots \quad \dfrac{{}^mx_n \not\simeq \bar{v}_n}{\bot \cdot ({}^mx_n = \bar{v}_n)} \quad \cdots \quad {}^mL_k}$$

$$\frac{\Gamma,\ (L_1 \vee \cdots \vee z \simeq u \vee \cdots \vee L_k),\ \Lambda \parallel \Delta, f(\vec{x}) \not\simeq y}{{}^mL_1 \quad \cdots \quad \bot \cdot (f(\vec{x}) = {}^mz > {}^mu = y) \quad \cdots \quad {}^mL_k} \quad \Longrightarrow$$

$$\frac{\Gamma,\ (L_1 \vee \cdots \vee z \simeq u \vee \cdots \vee L_k),\ \Lambda \parallel \Delta, f(\vec{x}) \not\simeq y}{{}^mL_1 \quad \cdots \quad \dfrac{\bar{w} \not\simeq y \cdot (f(\vec{x}) = {}^mz > \bar{w})}{\bot \cdot (\bar{w} = y)} \quad \dfrac{{}^mu \not\simeq \bar{w}}{\bot \cdot ({}^mu = \bar{w})} \quad \cdots \quad {}^mL_k}$$

$$\frac{\Gamma,\ (L_1 \vee \cdots \vee f(\vec{z}) \simeq u \vee \cdots \vee L_k),\ \Lambda \parallel \Delta, f(\vec{x}) \not\simeq y}{{}^mL_1 \quad \cdots \quad \bot \cdot (f(\vec{x}) = {}^mf(\vec{z}) > {}^mu = y) \quad \cdots \quad {}^mL_k} \quad \Longrightarrow$$

$$\frac{\Gamma,\ (L_1 \vee \cdots \vee f(\vec{z}) \simeq u \vee \cdots \vee L_k),\ \Lambda \parallel \Delta, f(\vec{x}) \not\simeq y}{{}^mL_1 \quad \cdots \quad \dfrac{\bar{w} \not\simeq y \cdot (f(\vec{x}) = f(\vec{v}) > \bar{w})}{\bot \cdot (\bar{w} = y)} \quad \dfrac{{}^mu \not\simeq \bar{w}}{\dfrac{\bot \cdot}{({}^mu = \bar{w})}} \quad \dfrac{{}^mz_1 \not\simeq \bar{v}_1}{\dfrac{\bot \cdot}{({}^mz_1 = \bar{v}_1)}} \quad \cdots \quad \dfrac{{}^mz_n \not\simeq \bar{v}_n}{\dfrac{\bot \cdot}{({}^mz_n = \bar{v}_n)}} \quad \cdots \quad {}^mL_k}$$

Рис. 12: Перетворення $\mathbf{CT}^{\simeq}$ у $\mathbf{LPCT}^{\simeq}$

- Змінна $\hat{u}$ не входить в $\mathbb{T}^{(0)}$ поза межами $\mathbb{T}^{\circ}$ та $\mathbb{T}^{\bullet}$.
- Літера в корені $\mathbb{T}^{\bullet}$ має вигляд $(s \not\simeq \hat{u})$, і $\hat{u}$ не входить в s.
- Взагалі, $\hat{u}$ зустрічається в $\mathbb{T}^{\bullet}$ лише в нерівностях виду $(t \not\simeq \hat{u})$ та в констрейнтах $(t = \hat{u})$, введених кроком редукції, причому $\hat{u}$ не входить в ці терми t (все, що ми можемо робити з нерівністю $(t \not\simeq \hat{u})$, це парамодулювати в t або закрити її кроком редукції).
- Змінна $\hat{u}$ зустрічається лише один раз в літері в корені $\mathbb{T}^{\circ}$.
- Змінна $\hat{u}$ не з'являється в констрейнті кореня $\mathbb{T}^{\circ}$.

Нехай $\mathbb{T}^{\circ}$ має вигляд:

$$\frac{M[\hat{u}] \cdot \delta}{\mathbb{T}_1 \quad \cdots \quad \mathbb{T}_n}$$

Далі, $\mathbb{T}_i[\hat{u} \leftarrow t]$ позначатиме табло $\mathbb{T}_i$, де всі входження $\hat{u}$ (як в літери, так і в констрейнти) замінені на терм t. Легко переконатися, що така підстановка не спотворює дерева $\mathbb{T}_i$, за умови, що $\hat{u}$ та t є рівними відносно констрейнтів в $\mathbb{T}^{(0)}$. Перетворення дерева $[T]^{\mathbb{T}^{\circ}}$ визначається наступним чином:

$$\left[\frac{t \not\simeq \hat{u} \cdot \gamma}{\bot \cdot (t = \hat{u})}\right]^{\mathbb{T}^{\circ}} \quad \Longrightarrow \quad \frac{M[t] \cdot \gamma}{\mathbb{T}_1[\hat{u} \leftarrow t] \quad \cdots \quad \mathbb{T}_n[\hat{u} \leftarrow t]}$$

$$\left[\frac{t \not\simeq \hat{u} \cdot \gamma}{T_1 \quad \cdots \quad T_n}\right]^{\mathbb{T}^{\circ}} \quad \Longrightarrow \quad \frac{M[t] \cdot \gamma}{[T_1]^{\mathbb{T}^{\circ}} \quad \cdots \quad [T_n]^{\mathbb{T}^{\circ}}}$$

$$\left[\frac{L \cdot \gamma}{T_1 \quad \cdots \quad T_n}\right]^{\mathbb{T}^{\circ}} \quad \Longrightarrow \quad \frac{L \cdot \gamma}{[T_1]^{\mathbb{T}^{\circ}} \quad \cdots \quad [T_n]^{\mathbb{T}^{\circ}}}$$

Розглянемо табло $[\mathbb{T}^{\bullet}]^{\mathbb{T}^{\circ}}$. Ми можемо стверджувати, що:

- Якщо $(s \not\simeq \hat{u})$ є літерою в корені $\mathbb{T}^{\bullet}$, то $M[s]$ є літерою в корені $[\mathbb{T}^{\bullet}]^{\mathbb{T}^{\circ}}$.
- Всі парамодуляції в літери $(t \not\simeq \hat{u})$, що були зроблені в $\mathbb{T}^{\bullet}$, залишаються коректними в $[\mathbb{T}^{\bullet}]^{\mathbb{T}^{\circ}}$. Справді, всі вони були зроблені в терм t, а змінилося лише оточення цього терму.

- Кожна літера ($t \not\simeq \hat{u}$), закрита редукцією в $\mathbb{T}^\bullet$ стає літерою $M[t]$ і продовжується піддеревами $\mathbb{T}_i[\hat{u} \leftarrow t]$. Оскільки $\hat{u}$ і t є рівними відносно констрейнтів з $\mathbb{T}^{(0)}$, дерево $[\mathbb{T}^\bullet]^{\mathbb{T}^\circ}$ є закритим.

Тепер ми видаляємо дерево $\mathbb{T}^\circ$ з $\mathbb{T}^{(0)}$ і замінюємо $\mathbb{T}^\bullet$ на $[\mathbb{T}^\bullet]^{\mathbb{T}^\circ} + \delta$ (тобто, δ додається до констрейнту в кореневому вузлі). Ми отримали правильно побудоване закрите табло $\mathbb{T}^{(1)}$, де змінна $\hat{u}$ не зустрічається. Неформально, $\mathbb{T}^{(1)}$ є деревом, яке було розширене не диз'юнктом $(\cdots \vee L[\hat{u}] \vee \cdots \vee p \not\simeq \hat{u} \vee \cdots)$, а диз'юнктом $(\cdots \vee L[p] \vee \cdots)$. Після розширення, ми робимо в терм p всі ті парамодуляції, що були зроблені в $\mathbb{T}^\bullet$, і продовжуємо з $L[\hat{u}]$ тими кроками виведення, що зроблені в $\mathbb{T}^\circ$.

Зауважимо, що в дереві $[\mathbb{T}^\bullet]^{\mathbb{T}^\circ}$ може з'явитися більш ніж одне піддерево вигляду $\mathbb{T}^\circ[\hat{u} \leftarrow q]$. Відповідно, змінні з кришечкою, що були введені всередині $\mathbb{T}^\circ$ можуть з'явитись кілька разів в піддеревах $\mathbb{T}^{(1)}$. Це не є проблемою, оскільки ми можемо елімінувати такі змінні, рухаючись від кореня до листів.

Повторюючи цю процедуру, ми отримаємо табло $\mathbb{T}^{(n)}$, де змінні з кришечкою не з'являються взагалі. Табло $\mathbb{T}^{(n)}$ є правильно побудованим закритим $\mathbf{LPCT}^{\simeq}$-табло, що побудоване з диз'юнктів з $\mathcal{S}$ після кроку елімінації симетрії в CEE. Множина констрейнтів в $\mathbb{T}^{(n)}$ випливає з сукупного констрейнту в $\mathbb{T}$, і отже є виконуваною.

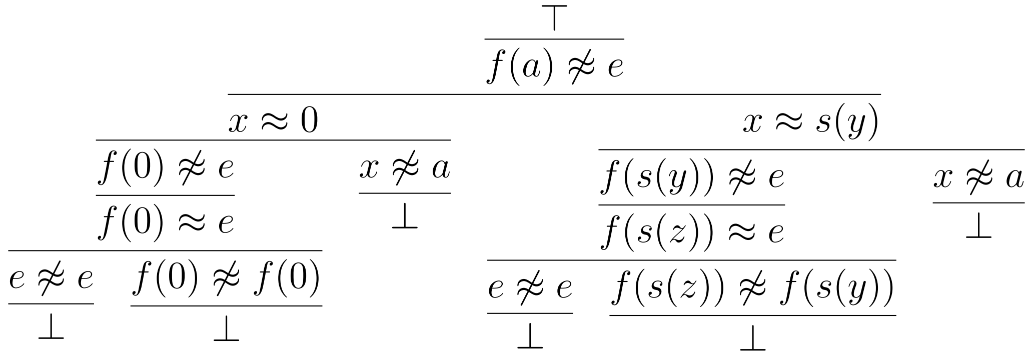
Залишається лише скасувати крок елімінації симетрії. Ми замінюємо символи $\simeq$ та $\not\simeq$ на, відповідно, $\approx$ та $\not\approx$, і переорієнтовуємо атоми рівності, де потрібно, до їхньої початкової форми в $\mathcal{S}$.

Ми отримали закрите $\mathbf{LPCT}$-табло, що спростовує $\mathcal{S}$. □

Незважаючи на спосіб доведення повности, числення $\mathbf{LPCT}$ не є простим переформулюванням методу CEE. Насправді, є істотня різниця між сплощенням термів і лінивою парамодуляцією. Змінні з кришечкою, введені в CEE-диз'юнктах, можуть розглядатися як "значення" термів, які вони заміщають. Інакше кажучи, терм, який буде підставлений в змінну $\hat{u}$, є, фактично, результатом всіх парамодуляцій, зроблених в терм t, що його замістила змінна $\hat{u}$ під час CEE-перетворення. Таким чином, в

даному CEE-диз'юнкті, кожен терм має в точності одне "значення". Для числення **LPCT** це не так.

Нехай $\mathcal{S}$ буде множиною $\{x \approx 0 \vee x \approx s(y),\ f(0) \approx e,\ f(s(z)) \approx e,\ f(a) \not\approx e\}$. Наступне табло, побудоване в численні з підрозділу 3.3.2 (ми беремо його для простоти, виведення в **LPCT** цілком аналогічне) неможливо отримати з будь-якого $\mathbf{CT}^{\simeq}$-спростування CEE($\mathcal{S}$).



В цьому виведенні ми заміщаємо константу a з початкового диз'юнкту двома різними термами, 0 та $s(y)$. Працюючи з CEE-диз'юнктами, ми повинні взяти два окремих екземпляра початкового диз'юнкту. На основі цього прикладу нескладно показати, що **LPCT** може дати експоненційне скорочення мінімального виведення, порівняно з $\mathbf{CT}^{\simeq}$.

ВИСНОВКИ

В дисертації автором запропоновано сукупність засобів формального подання математичних знань та їхньої автоматичної обробки з використанням методів природного математичного викладення та міркування. Ці засоби втілено в програмній системі обробки та верифікації формальних математичних текстів.

При виконанні роботи були одержані такі результати:

1. Розроблено формальну мову ForTheL для запису математичних текстів, близьку до природної англійської мови. Речення та розділи мови ForTheL можуть бути переведені в формули мови першого порядку. Семантика тексту в цілому визначається поставленим завданням (наприклад, перевірити коректність) і ґрунтується на відношенні логічного передування та процедурі нормалізації тексту, яка зводить доведення, проведені за різними схемами, до єдиного уніфікованого вигляду. Визначено різні рівні коректності ForTheL-тексту.

2. Сформульовано поняття локальної істинності твердження в заданій позиції всередині формули першого порядку. Розглянуто властивості цього поняття, зокрема, показано, що правило modus ponens може застосовуватись локальним чином. Доведена основна властивість локальної істинності: якщо еквівалентність двох формул є локально істинною в заданій позиції, ці дві формули є взаємозамінюваними в цій позиції. Поняття локальної істинності використовується у визначенні онтологічної коректності ForTheL-тексту, а також для теоретичного обґрунтування коректності методів доведення, що застосовують перетворення всередині формул.

3. Викладено табличне числення **GDT**, що використовує поняття допустимої підстановки та стратегію цілекерованості в стилі програми "Алгоритм Очевидності". Доведено коректність і повноту цього числення відносно класичного табличного диз'юнктного числення Model Elimination.

4. Запропоновано оригінальне цілекероване табличне числення з правилом лінивої парамодуляції **LPCT**, доведено його повноту.

5. Реалізовано програмну систему обробки та верифікації формальних математичних текстів, записаних у мові ForTheL. Ця система включає в себе процедури синтаксичного аналізу ForTheL-тексту, трансляції у внутрішнє нормалізоване подання, перевірки онтологічної та загальної коректності, генерації відомостей про окремі входження термів, розкриття визначень та спрощення складних цілей, а також комбінаторний прувер в класичній логіці першого порядку з рівністю, заснований на описаному у роботі цілекерованому табличному численні **GDT**.

В процесі досліджень автором було проведено серію експериментів з формалізації нетривіальних математичних текстів та їхньої верифікації в системі САД. Зокрема, було формалізовано наступні тексти:

1. доведення нескінченого та скінченого варіантів теореми Рамсея, а також принципу компактності [60];

2. доведення стабільності відношення уточнення (refinement) над класом специфікаций програм відносно деяких операцій над специфікаціями [61];

3. доведення деяких властивостей скінчених груп [62];

4. доведення збіжності ряду зі знакозмінним загальним членом, що монотонно спадає за абсолютним значенням;

5. доведення ірраціональності квадратного кореня з простого числа;

6. доведення нерівності Коши-Буняковського для векторів над $\mathbb{R}$.

Кожен з цих текстів було записано в мові ForTheL та перевірено на коректність системою САД. Міркування, що проводяться в цих текстах, містять доведення за індукцією, математичні конструкції другого порядку (границі та суми послідовностей, об'єднання сукупностей множин),

алгебраїчні та теоретико-множинні перетворення. В експериментах використовувався власний прувер системи САД, заснований на численні **GDT**, а також резолюційний прувер SPASS [38], розроблений в Саарбрюкенському університеті (Німеччина).

ПРЕДМЕТНИЙ ПОКАЖЧИК

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

[1] *Глушков В. М.* Некоторые проблемы теории автоматов и искусственного интеллекта // *Кибернетика.* — 1970. — № 2. — С. 3–13.

[2] *Глушков В. М.* Основания математики и автоматизация дедуктивных построений // Кибернетика. Вопросы теории и практики. — М.: Наука, 1986. — С. 463.

[3] A brief historical sketch on kiev school of automated theorem proving / A. Degtyarev, Y. Kapitonova, A. Letichevsky et al. // Proc. 2nd International THEOREMA Workshop. — Linz, Austria, 1998. — Pp. 151–156.

[4] О языке для описания формальных теорий / В. М. Глушков, В. Ф. Костырко, А. А. Летичевский и др. // *Теоретическая кибернетика.* — 1970. — № 3.

[5] О формальном языке для записи математических текстов / В. М. Глушков, К. П. Вершинин, Ю. В. Капитонова и др. // Автоматизация поиска доказательства теорем в математике. — Киев: ИК АН УССР, 1974. — С. 3–36.

[6] Об одном алгоритме поиска доказательства теорем в теории групп / Ф. В. Ануфриев, В. В. Федюрко, А. А. Летичевский и др. // *Кибернетика.* — 1966. — № 1. — С. 23–29.

[7] *Ануфриев Ф. В.* Алгоритм поиска доказательства теорем в логическом исчислении // *Теория автоматов.* — 1969. — № 5.

[8] *Дегтярев А. И., Лялецкий А. В.* Логический вывод в системе автоматизации доказательств // Математические основы систем искуственного интеллекта. — Киев: ИК АН УССР, 1981.

[9] *Лялецкий А.* Секвенциальный формализм и дедуктивные системы для классической логики 1-го порядка // Труды Междунар. конф. “Логика и приложения”. — Новосибирск, Россия: 2000.

[10] *Lyaletski A.* Gentzen calculi and admissible substitutions // Actes prelim. du Symp. Franco-Sovetigue "Informatika-91". — Grenoble, France: 1991. — Pp. 99–111.

[11] *Lyaletski A.* On Herbrand theorem // *The Bulletin of Symbolic Logic.* — 2001. — Vol. 7, no. 1. — Pp. 132–133.

[12] *Лялецький О. В., Паскевич А. Ю.* Про деякі стратегії пошуку логічного виводу, керовані цілями // *Вісник КНУ (фізико-математичні науки).* — 2001. — № 2. — С. 277–285.

[13] *Вершинин К. П., Лялецкий А. В., Паскевич А. Ю.* Применение Системы Автоматизации Дедукции для верификации математических текстов // *Международный научно-теоретический журнал "Искусственный интеллект".* — 2003. — № 3. — С. 57–69.

[14] *Vershinin K., Paskevich A.* ForTheL — the language of formal theories // Proc. International Conf. on Information Theories and Applications 2000. — Vol. 7(3) of *International Journal of Information Theories and Applications.* — Varna, Bulgary: 2000. — Pp. 120–126.

[15] *Lyaletski A., Verchinine K., Paskevich A.* On verification tools implemented in the System for Automated Deduction // Proc. Second CoLogNet Workshop on Implementation Technology for Computational Logic Systems. — Pisa, Italy: 2003. — Pp. 3–14.

[16] *Lyaletski A., Paskevich A., Verchinine K.* Theorem proving and proof verification in the system SAD // Mathematical Knowledge Management: Third International Conference, MKM 2004 / Ed. by A. Asperti, G. Bancerek, A. Trybulec. — Vol. 3119 of *Lecture Notes in Computer Science.* — Springer-Verlag, 2004. — Pp. 236–250.

[17] *Paskevich A.* Formalized theory language and mathematical knowledge processing // Proc. Inter. Workshop RTETP'2000. — Kyiv, Ukraine: 2000. — Pp. 54–61.

[18] *Lyaletski A., Paskevich A.* Goal-driven inference search in classical propositional logic // Proc. Inter. Workshop STRATEGIES'2001. — Siena, Italy: 2001. — Pp. 65–74.

[19] *Паскевич А. Ю.* Особливості реалізації мови високого рівня для запису математичних текстів // *Вісник КНУ (фізико-математичні науки).* — 1999. — № 2. — С. 284–288.

[20] *Паскевич А. Ю.* Поняття локальної істинності та його застосування у автоматичному доведенні теорем // *Вісник КНУ (фізико-математичні науки).* — 2003. — № 1. — С. 199–203.

[21] *Nipkow T., Paulson L. C., Wenzel M.* Isabelle/HOL: A Proof Assistant for Higher-Order Logic. — Springer-Verlag, 2002. — Vol. 2283 of *Lecture Notes in Computer Science.* — P. 218.

[22] The Coq proof assistant reference manual — version v6.1: Tech. Rep. 0203 / B. Barras, S. Boutin, C. Cornes et al.: INRIA, 1997.

[23] *Benzmüller C., Siekmann J. H. et al.* ΩMEGA: Towards a mathematical assistant // Automated Deduction: 14th International Conference, CADE-14 / Ed. by W. McCune. — Vol. 1249 of *Lecture Notes in Computer Science.* — Springer-Verlag, 1997. — Pp. 252–255.

[24] *Owre S., Rushby J. M., Shankar N.* PVS: a prototype verification system // Automated Deduction: 11th International Conference, CADE-11 / Ed. by D. Kapur. — Vol. 607 of *Lecture Notes in Computer Science.* — Springer-Verlag, 1992. — Pp. 748–752.

[25] *Gordon M. J. C., Melham T. F.* Introduction to HOL: a theorem proving environment for higher order logic. — Cambridge University Press, 1993. — P. 492.

[26] Selected Papers on Automath / Ed. by R. P. Nederpelt, J. H. Geuvers, R. C. de Vrijer. — North-Holland, 1994. — Vol. 133 of *Studies in Logic and the Foundations of Mathematics.* — Pp. xix + 1024.

[27] *Richardson J., Smaill A., Green I.* System description: Proof planning in higher-order logic with Lambda-Clam // Automated Deduction: 15th International Conference, CADE-15. — Vol. 1421 of *Lecture Notes in Computer Science.* — Springer-Verlag, 1998. — Pp. 129–133.

[28] *Lamport L.* Types considered harmful. — DEC SRC internal note.

[29] *Trybulec A., Blair H.* Computer assisted reasoning with Mizar // Proc. 9th International Joint Conference on Artificial Intelligence. — 1985. — Pp. 26–28.

[30] *Boyer R. S., Moore J. S.* A Computational Logic Handbook (2nd edition). — Academic Press, 1998. — P. 408.

[31] *Kaufmann M., Manolios P., Moore J. S.* Computer-Aided Reasoning: An Approach. — Kluwer Academic Publishers, 2000. — P. 292.

[32] *Buchberger B., Jebelean T. et al.* A survey of the Theorema project // ISSAC'97 — Proc. International Symposium on Symbolic and Algebraic Computation / Ed. by W. Küchlin. — Maui, Hawaii, USA: ACM Press, 1997. — Pp. 384–391.

[33] *Wenzel M.* Isar — a generic interpretative approach to readable formal proof documents // Theorem Proving in Higher Order Logics: 12th International Conference, TPHOLs'99. — Vol. 1690 of *Lecture Notes in Computer Science.* — Springer-Verlag, 1999. — Pp. 167–184.

[34] *Dixon L., Fleuriot J. D.* IsaPlanner: A prototype proof planner in Isabelle // Automated Deduction: 19th International Conference, CADE-19 / Ed. by F. Baader. — Vol. 2741 of *Lecture Notes in Computer Science.* — Springer-Verlag, 2003. — Pp. 279–283.

[35] Linguistic tools and deductive technique of the System for Automated Deduction / Z. Aselderov, K. Verchinine, A. Degtyarev et al. // Proc. 3rd International Workshop on the Implementation of Logics. — Tbilisi, Georgia: 2002. — Pp. 21–24.

[36] SAD, a System for Automated Deduction: a current state / K. Verchinine, A. Degtyarev, A. Lyaletski, A. Paskevich // Proc. Workshop on 35 Years of Automath / Ed. by F. Kamareddine. — Heriot-Watt University, Edinburgh, Scotland: 2002.

[37] *McCune W.* Otter 3.0 reference manual and guide: Tech. Report ANL-94/6. — Argonne, USA: Argonne National Laboratory, 1994.

[38] SPASS Version 2.0 / C. Weidenbach, U. Brahm, T. Hillenbrand et al. // Automated Deduction: 18th International Conference, CADE-18 / Ed. by A. Voronkov. — Vol. 2392 of *Lecture Notes in Computer Science*. — Springer-Verlag, 2002. — Pp. 275–279.

[39] *Riazanov A., Voronkov A.* The design and implementation of VAMPIRE // *AI Communications*. — 2002. — Vol. 15, no. 2–3. — Pp. 91–110.

[40] *Sutcliffe G., Suttner C. B., Yemenis T.* The TPTP problem library // Automated Deduction: 12th International Conference, CADE-12 / Ed. by A. Bundy. — Vol. 814 of *Lecture Notes in Computer Science*. — Springer-Verlag, 1994. — Pp. 252–266. — see also http://tptp.org.

[41] *Jones S. P.* Haskell 98 Language and Libraries. — Cambridge University Press, 2003. — P. 270.

[42] *Fuchs N. E., Schwertel U., Schwitter R.* Attempto Controlled English — not just another logic specification language // Logic-Based Program Synthesis and Transformation: 8th International Workshop, LOPSTR'98 / Ed. by P. Flener. — Vol. 1559 of *Lecture Notes in Computer Science*. — Springer-Verlag, 1999. — Pp. 1–20.

[43] *Sowa J. F.* Common Logic Controlled English (unpublished draft). — hosted at http://www.jfsowa.com/clce/specs.htm.

[44] *Фейс Р.* Модальная логика. — М.: Наука, 1974. — С. 520.

[45] *Monk L. G.* Inference rules using local contexts // *Journal of Automated Reasoning*. — 1988. — Vol. 4, no. 4. — Pp. 445–462.

[46] *Corella F.* What holds in a context? // *Journal of Automated Reasoning.* — 1993. — Vol. 10, no. 2. — Pp. 79–93.

[47] *Вершинин К. П.* Использование вспомогательных утверждений в процессе поиска доказательства // *Семиотика и Информатика.* — 1979. — № 12. — С. 12–21.

[48] *Kamareddine F., Nederpelt R. P.* A Refinement of de Bruijn's Formal Language of Mathematics // *Journal of Logic, Language and Information.* — 2004. — Vol. 13, no. 3. — Pp. 287–340.

[49] *Beth E. W.* Semantic entailment and formal derivability // Logik-Texte. Kommentierte Auswahl zur Geschichte der modernen Logik / Ed. by K. Berka, L. Kreiser. — Akademie-Verlag, Berlin, 1986. — Pp. 262–266.

[50] *Degtyarev A., Lyaletski A., Morokhovets M.* Evidence Algorithm and sequent logical inference search // Logic Programming and Automated Reasoning: 6th International Conference, LPAR'99 / Ed. by H. Ganzinger, D. A. McAllester, A. Voronkov. — Vol. 1705 of *Lecture Notes in Artificial Intelligence.* — Springer-Verlag, 1999. — Pp. 44–61.

[51] *Degtyarev A., Lyaletski A., Morokhovets M.* On the EA-style integrated processing of self-contained mathematical texts // Symbolic Computation and Automated Reasoning: The CALCULEMUS-2000 Symposium / Ed. by M. Kerber, M. Kohlhase. — A.K. Peters, Ltd., USA, 2001. — Pp. 126–141.

[52] System for Automated Deduction (SAD): Linguistic and deductive peculiarities / A. Lyaletski, K. Verchinine, A. Degtyarev, A. Paskevich // Intelligent Information Systems: IIS'2002 Symposium / Ed. by M. A. Klopotek, S. T. Wierzchon, M. Michalewicz. — Advances in Soft Computing. — Physica-Verlag, 2002. — Pp. 413–422.

[53] Handbook of Tableau Methods / Ed. by M. D'Agostino, D. Gabbay, R. Hähnle, J. Posegga. — Springer-Verlag, 1999. — P. 680.

[54] *Loveland D.* Mechanical theorem proving by model elimination // *Journal of the ACM.* — 1968. — Vol. 16, no. 3. — Pp. 349–363.

[55] *Letz R., Stenz G.* Model elimination and connection tableau procedures // Handbook for Automated Reasoning / Ed. by A. Robinson, A. Voronkov. — Elsevier Science, 2001. — Vol. II. — Pp. 2017–2116.

[56] *Degtyarev A., Voronkov A.* Equality reasoning in sequent-based calculi // Handbook for Automated Reasoning / Ed. by A. Robinson, A. Voronkov. — Elsevier Science, 2001. — Vol. I. — Pp. 609–704.

[57] *Brand D.* Proving theorems with the modification method // *SIAM Journal of Computing.* — 1975. — Vol. 4. — Pp. 412–430.

[58] *Bachmair L., Ganzinger H., Voronkov A.* Elimination of equality via transformation with ordering constraints // Automated Deduction: 15th International Conference, CADE-15 / Ed. by C. Kirchner, H. Kirchner. — Vol. 1421 of *Lecture Notes in Artificial Intelligence.* — Springer-Verlag, 1998. — Pp. 175–190.

[59] *Snyder W., Lynch C.* Goal directed strategies for paramodulation // Rewriting Techniques and Applications: 4th International Conference, RTA 1991. — Vol. 448 of *Lecture Notes in Computer Science.* — Springer-Verlag, 1991. — Pp. 150–161.

[60] *Graham R. L.* Rudiments of Ramsey Theory. — AMS, 1981. — P. 65.

[61] *Mammar A.* Un environnement formel pour le développement d'application bases de données: Ph.D. thesis / Conservatoire National des Arts et Métiers. — 2003. — P. 227.

[62] *Серр Ж.-П.* Курс арифметики. — М.: Мир, 1972.

[63] *Wiedijk F.* The Seventeen Provers of the World (unpublished draft). — hosted at http://www.cs.ru.nl/~freek/comparison/.

ДОДАТОК А
ФОРМАЛІЗАЦІЯ МАТЕМАТИЧНОГО ТЕКСТУ

Наприкінці розглянемо формалізацію в мові ForTheL математичного результату, який став класичним прикладом в галузі автоматичного доведення теорем [63].

Факт А.1. *Квадратний корень простого числа є ірраціональним.*

Нижче наведено повний текст ForTheL-формалізації цієї теореми. Окремі пояснення ми даємо в ForTheL-коментарі, які виділяються знаком '#'. Логічна коректність цього тексту (але не онтологічна коректність) автоматично перевірена системою САД з використанням пруверу SPASS.

```
#
# ЧАСТИНА I: БАЗОВІ ВІДОМОСТІ ПРО НАТУРАЛЬНІ ЧИСЛА
#
[a number/numbers] [x is natural] [the zero] [0 @ the zero]
[x is nonzero @ x is not equal to 0] [the one] [1 @ the one]
[x is nontrivial @ x != 0 /\ x != 1]

Axiom ZONat.   0 and 1 are natural numbers and 0 != 1.

[the sum of n and m] [n + m @ the sum of n and m]
[the product of n and m] [n * m @ the product of n and m]

Axiom AddNum.  For all numbers m,n (m + n) is a number.
Axiom AddNat.  For all natural numbers m,n (m + n) is natural.
Axiom _AddZero. For all numbers n (0 + n = n and n + 0 = n).

Axiom MulNum.  For all numbers m,n (m * n) is a number.
Axiom MulNat.  For all natural numbers m,n (m * n) is natural.
Axiom _MulZero. For all numbers n (0 * n = 0 and n * 0 = 0).
Axiom _MulUnit. For all numbers n (1 * n = n and n * 1 = n).

Axiom AddComm.  For all numbers m,n (m + n = n + m).
Axiom AddAsso.  For all numbers l,m,n (l + (m + n) = (l + m) + n).
Axiom MulComm.  For all numbers m,n (m * n = n * m).
```

```
Axiom MulAsso.  For all numbers l,m,n (l * (m * n) = (l * m) * n).

Axiom AddCanc.  Let n,m,l be numbers.
    If (l + n = l + m \/ n + l = m + l) then n = m.

Axiom MulCanc.  Let n,m,l be numbers.
    If (l * n = l * m \/ n * l = m * l) and l is nonzero then n = m.

Axiom AMDistr.  For all numbers l,m,n
    l * (m + n) = (l * m) + (l * n) and
    (m + n) * l = (m * l) + (n * l).

Axiom ZeroAdd.  For all numbers m,n if m + n = 0 then m = 0 and n = 0.
Lemma ZeroMul.  For all numbers m,n if m * n = 0 then m = 0 or n = 0.

[n is lesseq than m] [n <= m @ n is lesseq than m]
[the diff of m and n] [m - n @ the diff of m and n]
[n < m @ n <= m /\ n != m] [n > m @ m < n]

Definition DefLE.   Let n,m be numbers.
    n <= m iff for some number l (n + l = m).

Definition DefDiff.  Let m,n be numbers and m <= n.
    n - m is a number such that m + (n - m) = n.

Axiom NatDiff. Let m,n be natural numbers and m <= n.
    Then (n - m) is natural.

Lemma LERefl.   For every number n (n <= n).

Lemma LEAsym.   For all numbers n,m (n <= m /\ m <= n => n = m).
Proof.
    Let n,m be numbers such that (n <= m /\ m <= n).
    We have n + ((m - n) + (n - m)) = n + 0.
    Hence (m - n) = 0 and (n - m) = 0.
qed.

Lemma LETran.   For all numbers n,m,l (n <= m /\ m <= l => n <= l).
```

```
Axiom LETotal.  For all numbers n,m (n <= m \/ m < n).

Lemma MonAdd.  Let n,l,m be numbers and l < m.
    Then n + l < n + m and l + n < m + n.
Proof.
    (n + l) + (m - l) = n + m.
qed.

Lemma MonMul.   Let n,l,m be numbers.
    Assume that n is nonzero and l < m.
    Then n * l < n * m and l * n < m * n.
Proof.
    (n * l) + (n * (m - l)) = n * m (by AddComm, AMDistr).
qed.

Axiom LENTr.    For every natural number n (n = 0 \/ n = 1 \/ n > 1).

Lemma MonMul2.  Let n,l be natural numbers.
    If l is nonzero then n <= n * l.

###   Цієї аксиоми достатньо для проведення доведень по індукції ###
Axiom IH.   For all natural numbers n,m (n < m => n -<- m).

#
# ЧАСТИНА II: ВЛАСТИВОСТІ ВІДНОШЕННЯ ДІЛИМОСТІ
#
[n divides/divide m] [n | m @ n divides m] [n is dividing m @ n | m]
[a divisor/divisors of n @ a natural number dividing n]
[the quotient of n to m] [n / m @ the quotient of n to m]

Definition DefDiv. Let n,m be natural numbers.
    n divides m iff for some natural number p (m = n * p).

Definition DefQuot. Let n,m be natural numbers and m != 0 and m | n.
    n / m is a natural number such that
        (n / m) * m = n and m * (n / m) = n.

Lemma DivTrans. Let l,m,n be natural numbers.
    If l | m and m | n then l | n (by DefDiv,MulAsso).
```

```
Lemma DivSum. Let l,m,n be natural numbers.
    If l | m and l | n then l | m + n.
Proof.
    Suppose that l | m and l | n.
    Take a natural number p such that m = l * p.
    Take a natural number q such that n = l * q.
    Then m + n = l * (p + q) (by AMDistr).
    Hence the thesis.
qed.

Lemma DivMin. Let l,m,n be natural numbers.
    If l | m and l | m + n then l | n.
Proof.
    Suppose that l | m and l | m + n.
    Take a natural number p such that m = l * p.
    Take a natural number q such that m + n = l * q.
    Case q < p.
        Take a natural number r equal to p - q (by NatDiff).
        We have (l * q) = ((l * q) + (l * r)) + n (by AMDistr).
        Then (l * r) + n = 0 (by AddAsso, AddCanc). Hence n = 0.
    end.
    Case p <= q.
        Take a natural number r equal to q - p (by NatDiff).
        We have (l * p) + (l * r) = (l * p) + n (by AMDistr).
        Hence the thesis (by AddCanc).
    end.
qed.

Lemma DivLE. Let m,n be natural numbers.
    Assume that m | n and n is nonzero. Then m <= n.
Proof.
    Take a natural number q such that n = m * q.
    q is nonzero.
qed.

[x is prime] [x is compound @ x is not prime]

Definition DefPrime. Let n be a natural number.
```

```
    n is prime iff n is nontrivial and
        for every divisor m of n (m = 1 \/ m = n).

Lemma PrimNTR. Every natural prime number is nontrivial.

Lemma PrimDiv. Every nontrivial natural number k has a prime divisor.
Proof by induction.
    Let k be a nontrivial natural number.
    Case k is prime. Obvious.
    Case k is compound.
        Take a divisor q of k such that q != 1 and q != k.
        q is nontrivial and q < k.
        Take a prime divisor r of q.
        Then r divides k (by MulAsso).
    end.
qed.

#
# ЧАСТИНА III: ОСНОВНІ РЕЗУЛЬТАТИ
#
Lemma PDP.  For all natural numbers n,m,p
  if p is prime and p | n * m then p | n or p | m.
Proof by induction on ((n + m) + p).
  Let n,m,p be natural numbers.
  Assume that p is prime and p divides n * m.

  Case p <= n.
    Let us show that p divides (n - p) * m and n - p < n.
      n = p + (n - p) and n * m = p * ((n * m) / p).
      n * m = (p * m) + ((n - p) * m) (by AMDistr).
      p divides (n - p) * m (by DefDiv, DivMin).
    end.
    Then p divides n - p or p divides m (by IH).
    Indeed ((n - p) + m) + p < (n + m) + p (by MonAdd). end.
    If p divides n - p then p divides n (by DefDiv,DivSum).
  end.

  Case p <= m.
    Let us show that p divides n * (m - p) and m - p < m.
```

```
    m = p + (m - p) and n * m = p * ((n * m) / p).
    n * m = (p * n) + (n * (m - p)) (by MulComm,AMDistr).
    p divides n * (m - p) (by DefDiv, DivMin).
  end.
  Then p divides n or p divides m - p (by IH).
  Indeed (n + (m - p)) + p < (n + m) + p (by MonAdd). end.
  If p divides m - p then p divides m (by DefDiv,DivSum).
end.

Case n < p and m < p.
  Take a natural number k such that n * m = p * k.

  Case k = 0 \/ k = 1.
    If k = 1 then n = p or m = p (by MulComm).
  end.

  Case k != 0 /\ k != 1.
    Take a prime divisor r of k.
    Let us show that r divides n * m and r <= k and k < p.
      k = (k / r) * r and n * m = (p * (k / r)) * r.
      Assume that p <= k. Then we have k > m.
      p * k > p * m and p * m > n * m (by MonMul).
      Then p * k > n * m. We have a contradiction.
    end.
    Then r divides n or r divides m (by IH).
    Indeed (n + m) + r < (n + m) + p. end.

    Case r divides n.
      Let us prove that p divides (n / r) * m and (n / r) < n.
          k = (k / r) * r and n * m = (p * (k / r)) * r.
          n = (n / r) * r and n * m = ((n / r) * m) * r.
          Then p * (k / r) = (n / r) * m (by MulCanc).
          Hence p divides (n / r) * m (by DefDiv).
      end.
      Then p divides (n / r) or p divides m (by IH).
      Indeed ((n / r) + m) + p < (n + m) + p (by MonAdd). end.
      If p divides (n / r) then p divides n.
      Indeed (n / r) divides n (by DefDiv). end.
    end.
```

```
      Case r divides m.
        Let us prove that p divides n * (m / r) and (m / r) < m.
            k = (k / r) * r and n * m = (p * (k / r)) * r.
            m = (m / r) * r and n * m = (n * (m / r)) * r.
            Then p * (k / r) = n * (m / r) (by MulCanc).
            Hence p divides n * (m / r) (by DefDiv).
        end.
        Then p divides n or p divides (m / r) (by IH).
        Indeed (n + (m / r)) + p < (n + m) + p (by MonAdd). end.
        If p divides (m / r) then p divides m.
        Indeed (m / r) divides m (by DefDiv). end.
      end.
    end.
  end.
qed.

[x is relatively prime to y]
[x is rational] [the square of n @ n * n]

Definition DefRelPr. Let n,m be natural numbers.
    n,m are relatively prime iff n,m have no common nontrivial divisor.

Definition DefRat. Let q be a number.
    q is rational iff there exist natural relatively prime numbers n,m
        such that m is nonzero and q * m = n.

Theorem Main.  Let p be a natural prime number.
  For no rational number q the square of q is p.
Proof by contradiction.
  Let q be a rational number such that q * q = p.
  Take natural relatively prime numbers n,m such that q * m = n.
  Then p * (m * m) = (n * n) (by MulAsso,MulComm).
  Hence p divides n * n and p divides n.
  Choose a natural number k such that n = p * k.
  Then we have p * (m * m) = p * (k * n) (by MulAsso).
  The square of m is equal to (p * k) * k (by MulComm,MulCanc).
  Hence p divides m * m and p divides m (by MulAsso,DefDiv,PDP).
  We have a contradiction (by DefRelPr).
qed.
```