Рання історія Coq

Namdak Tonpa

Травень 2025

Анотація

Ця стаття присвячена ранній історії системи Coq — одного з найвпливовіших асистентів доведення у світі. Розглядаються логічні та теоретичні основи Числення індуктивних конструкцій, ключові етапи розвитку від перших прототипів до версій 7.x, а також внесок основних розробників: Тьєррі Кокана, Жерара Юе, Крістін Полен-Морінг та багатьох інших.

1 Рання історія Coq

1.1 Історичні корені

Coq — це асистент доведення для логіки вищого порядку, який дозволяє розробляти комп'ютерні програми, що відповідають їхнім формальним специфікаціям. Він є результатом приблизно десяти років досліджень проєкту Coq. Тут ми коротко розглянемо три основні аспекти: логічну мову, якою ми пишемо наші аксіоматизації та специфікації, асистент доведення, який дозволяє розробляти верифіковані математичні доведення, та екстрактор програм, який синтезує комп'ютерні програми, що відповідають їхнім формальним специфікаціям, записаним як логічні твердження в цій мові.

Логічна мова, яку використовує Coq, є різновидом теорії типів, відомою як Числення індуктивних конструкцій. Не повертаючись до Лейбніца та Буля, ми можемо датувати створення того, що зараз називається математичною логікою, працями Фреге та Пеано на межі століть. Виявлення антиномій при вільному використанні предикатів чи принципів розуміння спонукало Расселла обмежити предикатне числення стратифікацією типів. Ці зусилля завершилися створенням Principia Mathematica, першою систематичною спробою формального обґрунтування математики. Спрощення цієї системи за принципами просто типізованого λ-числення відбулося з Простою теорією типів Чьорча. Нотація λ-числення, спочатку використана для вираження функціональності, також могла застосовуватися як кодування доведень у природній дедукції. Ця ізоморфізм Керрі-Говарда використав Н. де Брюйн у проєкті Automath, першій повномасштабній спробі розробити та механічно верифікувати математичні доведення. Ці зусилля завершилися верифі-

кацією Юттінгом Grundlagen Ландау у 1970-х роках. Використовуючи цей ізоморфізм Керрі-Говарда, значні досягнення в теорії доведень привели до появи двох типо-теоретичних систем: перша, Інтуїціоністська теорія типів Мартіна-Льофа, намагається створити нову основу математики на конструктивних принципах; друга, поліморфне λ-числення Жирара, є дуже потужною функціональною системою, в якій ми можемо представляти структури доведень логіки вищого порядку. Поєднавши обидві системи у вищому порядку розширення мови Automath, Т. Кокан у 1985 році представив першу версію Числення конструкцій (CoC). Ця потужна логічна система дозволяла сильні аксіоматизації, але прямі індуктивні визначення були неможливі, і індуктивні поняття доводилося визначати опосередковано через функціональні кодування, що вводило неефективність і незручність. Формалізм був розширений у 1989 році Т. Коканом і К. Полен із додаванням примітивних індуктивних визначень, що призвело до сучасної Числення індуктивних конструкцій. Цей розширений формалізм тут не визначається строго. Натомість обговорюється багато конкретних прикладів. Зацікавленого читача відсилаємо до відповідних дослідницьких статей для отримання додаткової інформації про формалізм, його мета-теоретичні властивості та семантику. Однак розуміння цього теоретичного матеріалу не є необхідним для написання специфікацій. Можна зрозуміти Числення індуктивних конструкцій на вищому рівні як суміш предикатного числення, індуктивних визначень предикатів, представлених як типізований PROLOG, і рекурсивних визначень функцій, близьких до мови ML.

Автоматизоване доведення теорем було започатковане у 1960-х роках Девісом і Патнемом у пропозіційному численні. Повна механізація (у сенсі напіввирішувальної процедури) класичної логіки першого порядку була запропонована у 1965 році Дж.А. Робінсоном із єдиним уніфікованим правилом виведення, що називається резолюцією. Резолюція спирається на розв'язання рівнянь у вільних алгебрах (тобто структурах термів), використовуючи алгоритм уніфікації. У 1970-х роках досліджувалися численні вдосконалення резолюції, але мало переконливих реалізацій було створено, за винятком, звісно, того, що PROLOG певною мірою походить від цих зусиль. Менш амбітним підходом до розробки доведень є комп'ютерна перевірка доведень. Найвизначнішими перевірниками доведень, розробленими у 1970-х роках, були LCF, розроблений Р. Мілнером та його колегами в Единбурзькому університеті, спеціалізований на доведенні властивостей рекурсійних рівнянь денотаційної семантики, та теоремо-доведувач Бойєра і Мура, автоматизація примітивної рекурсії над індуктивними типами даних. У той час як теоремо-доведувач Бойєра-Мура намагався синтезувати доведення комбінацією автоматизованих методів, LCF будував свої доведення через програмування тактик, написаних у високорівневій функціональній мета-мові ML.

Відмінною рисою нашого асистента доведення, яка чітко відрізняє його, наприклад, від LCF або доведувача Бойєра і Мура, є можливість витягувати програми з конструктивного змісту доведень. Ця обчислювальна інтерпре-

тація об'єктів доведень, у традиції конструктивної математики Бішопа, базується на інтерпретації реалізовності у сенсі Кліні, завдяки К. Полен. Користувач повинен лише позначити свій намір, розділяючи в логічних твердженнях твердження про існування обчислювального об'єкта від логічних тверджень, які визначають його властивості, але які можуть розглядатися як просто коментарі у відповідній програмі. Маючи цю інформацію, система автоматично витягує функціональний терм із доведення консистентності його специфікацій. Цей функціональний терм, у свою чергу, може бути скомпільований у реальну комп'ютерну програму. Ця методологія витягування програм із доведень є революційною парадигмою для інженерії програмного забезпечення. Синтез програм давно був темою досліджень у штучному інтелекті, започаткованих Р. Валдінгером. Система Tablog З. Манна та Р. Валдінгера дозволяє дедуктивний синтез функціональних програм із доведень у табличній формі їхніх специфікацій, записаних у різновиді логіки першого порядку. Розробка систематичної логіки програмування, заснованої на розширеннях теорії типів Мартіна-Льофа, була здійснена в Корнельському університеті командою Nuprl на чолі з Р. Констеблем. Перший реальний екстрактор програм, PX, був розроблений і реалізований близько 1985 року С. Хаяші з Кіотського університету. Він дозволяє витягувати програму на LISP із доведення в логічній системі, натхненній логічними формалізмами С. Фефермана. Інтерес до цієї методології зростає в теоретичній комп'ютерній науковій спільноті. Ми можемо передбачити день, коли реальні комп'ютерні системи, що використовуються в додатках, міститимуть сертифіковані модулі, автоматично згенеровані з доведення консистентності їхніх формальних специфікацій. Однак ми все ще далекі від того, щоб використовувати цю методологію в плавній взаємодії зі стандартними інструментами інженерії програмного забезпечення, такими як компілятори, лінкери, системи виконання, що використовують спеціальне апаратне забезпечення, дебагери тощо. Ми сподіваємося, що Coq може бути корисним для дослідників, зацікавлених в експериментах із цією новою методологією.

1.2 Версії з 1 по 5

```
Цей огляд був написаний у 1995 році
разом із попереднім розділом
і складав початкову версію розділу "Кредити".

Більш детальний опис цих ранніх версій
доступний у наступних підрозділах,
які взяті з документа, написаного
у вересні 2015 року Жераром Юе,
Тьєррі Коканом і Крістін Полен.
```

Перша реалізація CoC була розпочата у 1984 році Ж. Юе та Т. Коканом. Мовою реалізації був CAML, функціональна мова програмування з сімей-

ства ML, розроблена в INRIA в Роккенкурі. Ядром цієї системи був перевірник доведень для CoC, розглянутий як типізоване λ-числення, назване Конструктивним двигуном. Цей двигун управлявся через високорівневу нотацію, що дозволяла оголошення аксіом і параметрів, визначення математичних типів і об'єктів, а також явну побудову об'єктів доведень, закодованих як λ-терми. Механізм секцій, розроблений і реалізований Ж. Довеком, дозволяв ієрархічний розвиток математичних теорій. Ця високорівнева мова називалася Математичним народним діалектом. Крім того, інтерактивний теоремо-доведувач дозволяв поступове побудування дерев доведень зверху вниз, рекурсивно визначаючи підцілі та відступаючи від тупиків. Теоремо-доведувач виконував тактики, написані на CAML, у стилі LCF. Базовий набір тактик був попередньо визначений, який користувач міг розширити своїми специфічними тактиками. Ця система (версія 4.10) була випущена у 1989 році. Потім система була розширена для роботи з новим численням із індуктивними типами К. Полен, з відповідними новими тактиками для доведень за індукцією. Був упорядкований новий стандартний набір тактик, а народний діалект розширений для виконання тактик. Пакет для компіляції програм, витягнутих із доведень, у реальні комп'ютерні програми на CAML або іншій функціональній мові був розроблений і реалізований Б. Вернером. Новий користувацький інтерфейс, що базується на інтерфейсі CAML-X від Д. де Роглодра, був розроблений і реалізований А. Фелті. Він дозволяв керувати теоремо-доведувачем через маніпуляцію вікнами, меню, кнопками, чутливими до миші, та іншими віджетами. Ця система (версія 5.6) була випущена у 1991 році.

Coq був перенесений на нову реалізацію Caml-light від К. Леруа та Д. Доліже в 1992 році Д. де Роглодром (версія 5.7). Нова версія Coq потім координувалася К. Мурті, з новими інструментами, розробленими К. Паран для доведення властивостей програм на ML (ця методологія є дуальною до витягування програм) і новим циклом взаємодії з користувачем. Ця система (версія 5.8) була випущена у травні 1993 року. Інтерфейс Centaur CTCoq потім був розроблений І. Берто з проєкту Croap від INRIA-Софія-Антиполіс.

Паралельно Ж. Довек і Х. Хербелін розробили новий двигун доведень, що дозволяє загальну маніпуляцію екзистенційними змінними, узгодженими з залежними типами, в експериментальній версії Coq (V5.9).

Версія V5.10 Coq базується на загальній системі для маніпуляції термами зі зв'язуючими операторами, розробленій Четом Мурті. Новий двигун доведень дозволяє паралельний розвиток часткових доведень для незалежних підцілей. Структура цих дерев доведень є змішаною репрезентацією дерев виведення для Числення індуктивних конструкцій з абстрактними синтаксичними деревами для скриптів тактик, дозволяючи навігацію в доведенні на різних рівнях деталізації. Двигун доведень дозволяє керувати загальними елементами середовища об'єктно-орієнтованим способом. Ця нова архітектура, завдяки Ч. Мурті, підтримує кілька нових можливостей, які полегшують розширення та масштабування системи:

Дозволені тактики, програмовані користувачем;

Можна окремо верифікувати модулі розробки та завантажувати їх скомпільовані образи без повторної верифікації — швидкий процес переміщення дозволяє їх швидке завантаження;

Загальна схема розбору дозволяє користувацькі нотації з симетричним табличнокерованим красивим друком;

Синтаксичні визначення дозволяють зручні скорочення;

Обмежена можливість мета-змінних дозволяє автоматичний синтез певних типових виразів, дозволяючи загальні нотації, наприклад, для рівності, парного зв'язування та екзистенційного квантифікатора.

Восени 1994 року К. Полен-Морінг замінила структуру індуктивно визначених типів і сімейств на нову структуру, що дозволяє взаємно рекурсивні визначення. П. Манурі реалізував переклад рекурсивних визначень у примітивний рекурсивний стиль, накладений внутрішніми операторами рекурсії, у стилі системи ProPre. К. Муньос реалізував процедуру прийняття рішень для інтуїціоністської пропозиційної логіки, базуючись на результатах Р. Діккгоффа. Ж.К. Фільятр реалізував процедуру прийняття рішень для логіки першого порядку без скорочення, базуючись на результатах Дж. Кетонена та Р. Вейрауха. Нарешті, Ч. Мурті реалізував бібліотеку тактик інверсії, звільняючи користувача від нудних визначень "предикатів інверсії".

Роккенкур, 1 лютого 1995 року
Жерар Юе

1.2.1 Версія 1

Ця програма є прототипом перевірника типів для логічного формалізму вищого порядку, відомого як Теорія конструкцій, представленого в докторській дисертації Тьєррі Кокана, з впливами системи F Жирара та Automath де Брюйна. Математичний аналіз системи є докторською роботою Тьєррі Кокана. Програма в основному є роботою Жерара Юе. Більшість математичних прикладів, верифікованих за допомогою програмного забезпечення, належать Тьєррі Кокану.

Мовою програмування програмного забезпечення CONSTR (як воно тоді називалося) була версія ML, адаптована з системи Edinburgh LCF і запущена на базі LISP. Основні вдосконалення порівняно з оригінальним LCF ML полягали в тому, що ML компілювався, а не інтерпретувався (Жерар Юе розвинув оригінальний транслятор Локвуда Морріса), і був збагачений рекурсивно визначеними типами (робота Гі Кузіно). Цей предок CAML використовувався та вдосконалювався Ларрі Полсоном для його реалізації Cambridge LCF.

Розробка цього прототипу відбувалася з кінця 1983 року до початку 1985 року.

Версія 1.10 була заморожена 22 грудня 1984 року. Це версія, використана для прикладів у дисертації Тьєррі Кокана, захищеній 31 січня 1985 року. Існував єдиний зв'язуючий оператор, який використовувався як для універсального квантифікатора (залежний добуток) на рівні типів, так і для функціональної абстракції (λ) на рівні термів/доведень, у стилі Automath. Підстановка (λ-редукція) була реалізована з використанням індексів де Брюйна.

Версія 1.11 була заморожена 19 лютого 1985 року. Це версія, використана для прикладів у статті: Т. Кокан, Ж. Юе. Constructions: A Higher Order Proof System for Mechanizing Mathematics [CH85].

Крістін Полен приєдналася до команди на цьому етапі для свого дослідницького стажування DEA. У своєму звіті DEA (серпень 1985) вона представляє розробки для функції lambo — обчислює мінімальне значення, таке що певна зростаюча цілочисельна функція перевищує задане значення, що є викликом для конструктивної математики. Вона також закодувала алгоритм голосування за більшістю Бойєра і Мура.

1.2.2 Версія 2

Формальна система, тепер перейменована на Числення конструкцій, була представлена з доведенням консистентності та порівнянням із системами доведень Пера Мартіна-Льофа, Жирара та сімейства Automath Н. де Брюйна в статті: Т. Кокан і Ж. Юе. The Calculus of Constructions [CH86b].

Абстракція дизайну програмного забезпечення у вигляді абстрактної машини для перевірки доведень і повніша послідовність математичних розробок були представлені в: Т. Кокан, Ж. Юе. Concepts Mathématiques et Informatiques Formalisés dans le Calcul des Constructions [CH86a].

Версія 2.8 була заморожена 16 грудня 1985 року і використовувалася для розробки прикладів у вищезгаданих статтях.

Це числення потім було збагачене у версії 2.9 кумулятивною ієрархією універсумів. Рівні універсумів спочатку були явними натуральними числами. Іншим покращенням стала можливість автоматичного синтезу неявних аргументів типів, що звільняло користувача від нудних повторюваних оголошень.

Крістін Полен написала статтю Algorithm development in the Calculus of Constructions [Moh86]. Окрім lambo та majority, вона представляє quicksort і алгоритм форматування тексту.

Версія 2.13 Числення конструкцій з універсумами була заморожена 25 червня 1986 року.

Синтетичне представлення теорії типів у конструктивному напрямку з алгоритмами ML було подано Жераром Юе у його нотатках до курсу CMU у травні 1986 року Formal Structures for Computation and Deduction. Його

розділ Induction and Recursion in the Theory of Constructions був представлений як запрошена доповідь на Спільній конференції з теорії та практики розробки програмного забезпечення TAPSOFT'87 у Пізі в березні 1987 року і опублікований як Induction Principles Formalized in the Calculus of Constructions [Hue88].

1.2.3 Версія 3

Ця версія побачила початок автоматизації доведень із пошуковим алгоритмом, натхненним PROLOG і програмами логічного програмування з курсових нотаток Formal structures for computation and deduction. Алгоритм пошуку був реалізований у ML Тьєррі Коканом. Таким чином, система доведень могла використовуватися у двох режимах: верифікація доведень і синтез доведень, з тактиками, такими як AUTO.

Мова реалізації тепер називалася CAML, для Categorical Abstract Machine Language. Як бекенд використовувалася віртуальна машина LLM3 від Le Lisp Жерома Шаю. Основними розробниками CAML були Мішель Монні, Аскандер Суарез і П'єр Вейс.

V3.1 була розпочата влітку 1986 року, V3.2 заморожена наприкінці листопада 1986 року. V3.4 розроблялася в першій половині 1987 року.

Тьєррі Кокан обіймав постдокторську посаду в Кембриджському університеті в 1986-87 роках, де він розробив варіант реалізації в SML, з яким написав деякі розробки щодо фікс-пойнтів у доменах Скотта.

1.2.4 Версія 4

Ця версія побачила початок витягування програм із доведень із двома різновидами типу Prop для пропозицій, що вказують на конструктивний намір. Алгоритми витягування доведень були реалізовані Крістін Полен-Морінг.

V4.1 була заморожена 24 липня 1987 року. Вона мала першу ідентифіковану бібліотеку математичних розробок (каталог exemples), з бібліотеками Logic (що містила імпредикативні кодування інтуїціоністської логіки та алгебраїчні примітиви для булевих значень, натуральних чисел і списків), Peano, що розвивала арифметику Пеано другого порядку, Arith, що визначала додавання, множення, евклідів поділ і факторіал. Типовими розробками були теорема Кнастера-Тарського та лема Ньюмена з теорії переписування.

V4.2 була спільною розробкою команди, що складалася з Тьєррі Кокана, Жерара Юе та Крістін Полен-Морінг. Файл V4.2.log фіксує журнал змін. Вона була заморожена у вересні 1987 року як остання версія, реалізована в CAML 2.3, а V4.3 пішла за CAML 2.5, більш стабільною системою розробки.

V4.3 побачила перший верхній рівень системи. Замість оцінки явних цитат користувач міг розробляти свою математику у високорівневій мові, названій

математичним народним діалектом (за термінологією Automath). Користувач міг розробляти файли в нотації народного діалекту (з розширенням .v), які тепер були відокремлені від джерел ml реалізації. Жіль Довек приєднався до команди для розробки мови народного діалекту як дослідження для свого стажування DEA.

Була введена концепція липкої константи, щоб зберігати назви лем під час зняття локальних гіпотез доведень. Це дало поняття глобального математичного середовища з локальними секціями.

Ще одна значна практична зміна полягала в тому, що система, спочатку розроблена на центральному комп'ютері VAX нашого лабораторії, була перенесена на персональні робочі станції SUN, що дозволило рівень розподіленої розробки. Алгоритм витягування був модифікований, із трьома анотаціями Pos, Null і Typ, що прикрашали сорти Prop і Type.

Версія 4.3 була заморожена наприкінці листопада 1987 року і була розповсюджена серед ранньої спільноти користувачів (серед них були Х'юго Хербелін і Лоїк Колсон).

V4.4 побачила першу версію (закодованих) індуктивних типів. Тепер натуральні числа могли бути визначені як:

```
Inductive NAT : Prop = O : NAT | Succ : NAT->NAT.
```

Ці індуктивні типи були закодовані імпредикативно в численні, використовуючи підсистему rec завдяки Крістін Полен. V4.4 була заморожена 6 березня 1988 року.

Версія 4.5 була першою, що підтримувала індуктивні типи та витягування програм. Її гаслом було Calcul des Constructions avec Réalisations et Synthèse. Мова народного діалекту була збагачена для розміщення команд витягування.

Дизайн двигуна верифікації був представлений як: Ж. Юе. The Constructive Engine. Version 4.5. Запрошена конференція, 2-й Європейський симпозіум із програмування, Нансі, березень 88. Остаточна стаття, що описує реалізацію V4.9, з'явилася в: A perspective in Theoretical Computer Science, Commemorative Volume in memory of Gift Siromoney, ред. Р. Нарасімхан, World Scientific Publishing, 1989.

Версія 4.5 була продемонстрована у червні 1988 року на Інституті YoP з логічних основ функціонального програмування, організованому Жераром Юе в Остіні, Техас.

Версія 4.6 була розпочата влітку 1988 року. Її основним покращенням була повна переробка двигуна синтезу доведень Тьєррі Коканом із деревовидною структурою цілей.

Її вихідний код був переданий Ренді Поллаку 2 вересня 1988 року. Він поступово еволюціонував у LEGO, систему доведень для формалізму Луо Роз-

ширеного числення конструкцій.

Тактика зняття була модифікована Жераром Юе, щоб дозволити взаємозалежності у знятих лемах. Крістін Полен покращила схему індуктивного визначення, щоб розмістити предикати будь-якої арності.

Версія 4.7 була розпочата 6 вересня 1988 року.

Ця версія починає використовувати поняття модуля CAML для покращення модульності реалізації. Тепер верифікатор термів ідентифікується як належний модуль Machine, структура внутрішніх даних якого прихована і доступна лише через легітимні операції. Ця машина (конструктивний двигун) була довіреним ядром реалізації. Механізм синтезу доведень був окремим генератором термів доведень. Після того, як повний терм доведення був синтезований за допомогою тактик, він повністю перевірявся двигуном. Таким чином, не було потреби сертифікувати тактики, і система скористалася цим фактом, дозволяючи тактикам ігнорувати рівні універсумів, перевірка консистентності універсумів відкладалася на останній прохід перевірки типів. Це викликало певне здивування у ранніх користувачів, які бачили після успішного пошуку доведення своє QED, за яким слідувала тиша, а потім повідомлення про помилку через невідповідність універсумів...

Набір прикладів включав експерименти з теорією множин Х'юго Хербеліна, зокрема теорему Шредера-Бернштейна.

Версія 4.8, розпочата 8 жовтня 1988 року, побачила значну переробку типу абстрактного синтаксису constr, відокремлюючи змінні формалізму та мета-змінні, що позначають неповні терми, керовані механізмом пошуку. Поняття рівня (з трьома значеннями TYPE, OBJECT і PROOF) стало явним, а судження типу прояснює конструкції, реалізація яких тепер повністю явна. Структурна рівність прискорюється за допомогою рівності вказівників, що дало вражаючі покращення. Тьєррі Кокан адаптує синтез доведень до нової репрезентації і спрощує зіставлення зразків до зіставлення предикатного числення першого порядку, з значним приростом продуктивності.

Нова репрезентація ієрархії універсумів була потім визначена Жераром Юе. Рівні універсумів тепер реалізовані неявно через прихований граф абстрактних рівнів, обмежених відношенням порядку. Перевірка ациклічності графу забезпечує обґрунтованість порядкування, а отже, консистентність. Це було задокументовано в меморандумі Adding Type:Type to the Calculus of Constructions, який так і не був опублікований.

Розробницька версія була випущена як стабільна 4.8 наприкінці 1988 року.

Версія 4.9 була випущена 1 березня 1989 року з новою "еластичною" ієрархією універсумів.

Весна 1989 року побачила першу спробу документувати використання системи, з низкою статей, що описують формалізм:

Metamathematical Investigations of a Calculus of Constructions, Тьєррі Кокан [Coq89],

Inductive definitions in the Calculus of Constructions, Крістін Полен-Морін,

Extracting Fω's programs from proofs in the Calculus of Constructions, Крістін Полен-Морінг [PM89],

The Constructive Engine, Жерар Юе [Hue89],

а також низка посібників для користувачів:

A short user's guide for the Constructions, Version 4.10, Жерар Юе,

A Vernacular Syllabus, Жіль Довек,

The Tactics Theorem Prover, User's guide, Version 4.10, Тьєррі Кокан.

Стабільна V4.10, випущена 1 травня 1989 року, була тоді зрілою системою, розповсюдженою з CAML V2.6.

Тим часом Тьєррі Кокан і Крістін Полен-Морінг досліджували, як додати нативні індуктивні типи до Числення конструкцій у стилі Інтуїціоністської теорії типів Пера Мартіна-Льофа. Імпредикативне кодування вже було представлено в: Ф. Пфеннінг і К. Полен-Морінг. Inductively defined types in the Calculus of Constructions [PPM89]. Розширення числення з примітивними індуктивними типами з'явилося в: Т. Кокан і К. Полен-Морінг. Inductively defined types [CP90].

Це призвело до Числення індуктивних конструкцій, логічного формалізму, реалізованого у версіях 5 і вище системи, і задокументованого в: К. Полен-Морінг. Inductive Definitions in the System Coq - Rules and Properties [PM93b].

Остання версія CONSTR — це версія 4.11, яка востаннє розповсюджувалася навесні 1990 року. Вона була продемонстрована на першому семінарі Європейської базової дослідницької дії Logical Frameworks у Софії-Антиполіс у травні 1990 року.

1.2.5 Версія 5

Наприкінці 1989 року була розпочата версія 5.1 і перейменована в систему Coq для Числення індуктивних конструкцій. Потім вона була перенесена на нову автономну реалізацію ML, що називалася Caml-light.

У 1990 році відбулося багато змін. Тьєррі Кокан поїхав до Університету Чалмерса в Гетеборзі. Крістін Полен-Морінг обійняла посаду дослідника CNRS у лабораторії LIP Вищої нормальної школи Ліона. Проєкт Formel був завершений і дав початок двом командам: Cristal в INRIA-Роккенкур, яка продовжила розробки в функціональному програмуванні з Caml-light,

а потім OCaml, і Coq, що продовжила дослідження теорії типів, із спільною командою під керівництвом Жерара Юе в INRIA-Роккенкур і Крістін Полен-Морінг у лабораторії LIP CNRS-ENS Ліон.

Четан Мурті приєднався до команди у 1991 році і став головним архітектором програмного забезпечення версії 5. Він повністю переробив реалізацію для підвищення ефективності. Версії 5.6 і 5.8 були основними розповсюдженими версіями, з повною документацією та бібліотекою розробок користувачів. Використання системи контролю версій RCS і систематичних файлів ChangeLog дозволяють більш точно відстежувати розробки програмного забезпечення.

Вересень 2015 року
Тьєррі Кокан, Жерар Юе і Крістін Полен-Морінг.

1.3 Версії 6

1.3.1 Версія 6.1

Поточна версія 6.1 Coq базується на архітектурі V5.10. Вона була перенесена на нову мову Objective Caml Бруно Баррасом. Основна структура дещо змінилася і дозволяє більше перетворень між сортами.

Нова версія надає потужні інструменти для легшого розвитку.

Крістіна Корнес розробила розширення синтаксису Coq, щоб дозволити визначення термів за допомогою потужного аналізу зіставлення зразків у стилі програм ML.

Амокран Саїбі написав механізм для симуляції успадкування між сімействами типів, розширюючи пропозицію Пітера Акселя. Він також розробив механізм для автоматичного обчислення аргументів константи, які система може вивести, і, відповідно, не потребують явного запису.

Ян Коской розробив команду, яка пояснює терм доведення природною мовою. П'єр Крегут створив нову тактику, яка розв'язує задачі в арифметиці Пресбургера без кванторів. Обидві функціональності були інтегровані в систему Coq Х'юго Хербеліном.

Самуель Бутен розробив тактику для спрощення комутативних кілець, використовуючи канонічний набір правил переписування та рівність модуля асоціативності та комутативності.

Нарешті, організація дистрибуції Coq була під наглядом Жан-Крістофа Фільятра за допомогою Жюдікаеля Курана і Бруно Барраса.

Ліон, 18 листопада 1996 року
Крістін Полен

1.3.2 Версія 6.2

У версії 6.2 Coq розбір виконується за допомогою camlp4, препроцесора і красивого друкаря для CAML, розробленого Даніелем де Роглодром в INRIA. Даніель де Роглодр зробив першу адаптацію Coq для camlp4, цю роботу продовжив Бруно Баррас, який також змінив структуру абстрактних синтаксичних дерев Coq і примітиви для їх маніпуляції. Результатом цих змін стала швидша процедура розбору з значно покращеними повідомленнями про синтаксичні помилки. Користувацький інтерфейс для введення правил граматики або красивого друку також змінився.

Едуардо Хіменес переробив внутрішні бібліотеки тактик, надавши однакові імена функціям Caml, що відповідають назвам тактик Coq.

Бруно Баррас написав нові, більш ефективні функції редукції.

Х'юго Хербелін ввів більш уніфіковані нотації в мові специфікацій Coq: визначення через фікс-пойнти та зіставлення зразків отримали більш читабельний синтаксис. Патрік Луазелер ввів зручні для користувача нотації для арифметичних виразів.

Були введені нові тактики: Едуардо Хіменес покращив механізм введення макросів для тактик і розробив спеціальні тактики для (ко)індуктивних визначень; Патрік Луазелер розробив тактику для спрощення поліноміальних виразів у довільному комутативному кільці, яка узагальнює попередню тактику, реалізовану Самуелем Бутеном. Жан-Крістоф Фільятр ввів тактику для уточнення цілі, використовуючи терм доведення з дірками як схему доведення.

Давид Делай розробив інструмент для пошуку об'єкта в бібліотеці за його типом (до ізоморфізму).

Анрі Лаульєр створив дистрибуцію Coq для середовища Windows.

Нарешті, Х'юго Хербелін був головним координатором документації Coq з основними внесками Бруно Барраса, Давида Делая, Жан-Крістофа Фільятра, Едуардо Хіменеса, Х'юго Хербеліна та Патріка Луазелера.

Орсе, 4 травня 1998 року
Крістін Полен

1.3.3 Версія 6.3

Основними змінами у версії V6.3 стали введення кількох нових тактик і розширення граничних умов для визначень фікс-пойнтів.

Б. Баррас розширив алгоритм уніфікації для завершення часткових термів і виправив різні складні помилки, пов'язані з універсумами.

Д. Делай розробив тактику AutoRewrite. Він також розробив нову поведінку Intro і надав тактикали First і Solve.

Ж.-К. Фільятр розробив тактику Correctness.

Е. Хіменес розширив умову охорони у фікс-пойнтах.

Х. Хербелін розробив нову синтаксису для визначень і розширив тактику Induction.

П. Луазелер розробив тактику Quote і новий дизайн тактики Auto, він також ввів індекс помилок у документацію.

К. Полен написала команду Focus і ввела функції редукції у визначеннях, ця остання функція була запропонована Ж.-Ф. Моніном із CNET Lannion.

Орсе, грудень 1999 року
Крістін Полен

1.4 Версії 7

1.4.1 Огляд змін

Версія V7 — це нова реалізація, розпочата у вересні 1999 року Жан-Крістофом Фільятром. Це значна ревізія внутрішньої архітектури системи. Версія Coq 7.0 була розповсюджена у березні 2001 року, версія 7.1 у вересні 2001 року, версія 7.2 у січні 2002 року, версія 7.3 у травні 2002 року і версія 7.4 у лютому 2003 року.

Жан-Крістоф Фільятр розробив архітектуру нової системи. Він ввів нову репрезентацію для середовищ і написав нове ядро для перевірки типів термів. Його підхід полягав у використанні функціональних структур даних для отримання більшого спільного використання, підготовки до додавання модулів і наближення до сертифікованого ядра.

Х'юго Хербелін ввів нову структуру термів із локальними визначеннями. Він ввів "кваліфіковані" імена, написав новий алгоритм компіляції зіставлення зразків і розробив більш компактний алгоритм для перевірки логічної консистентності універсумів. Він сприяв спрощенню внутрішніх структур Coq і оптимізації системи. Він додав базові тактики для прямого міркування і примушення в зразках.

Давид Делай ввів нову мову для тактик. Загальні тактики, що використовують зіставлення зразків на цілях і контексті, можуть бути написані безпосередньо з верхнього рівня Coq. Він також надав примітиви для розробки користувацьких тактик у Caml.

Мікаела Майєро внесла бібліотеку з дійсними числами. Олів'є Десметр розширив цю бібліотеку аксіоматичними тригонометричними функціями, квадратом, квадратними коренями, скінченними сумами, властивістю Шаля і базовою планіметрією.

Жан-Крістоф Фільятр і П'єр Летузі переробили нову процедуру витягування з термів Coq у програми на Caml або Haskell. Ця нова процедура

витягування, на відміну від попередньої, реалізованої в попередніх версіях Coq, здатна обробляти всі терми в Калькуляції індуктивних конструкцій, навіть ті, що включають універсуми і сильну елімінацію. П. Летузі адаптував внески користувачів для витягування програм ML, коли це було доцільно. Жан-Крістоф Фільятр написав coqdoc, інструмент документації для бібліотек Coq, який використовується з версії 7.2.

Бруно Баррас покращив ефективність алгоритму редукції і рівень довіри до правильності критичного алгоритму перевірки типів Coq.

Ів Берто розробив інструменти SearchPattern і SearchRewrite та підтримку для інтерфейсу pcoq[](http://www-sop.inria.fr/lemme/pcoq/).

Мікаела Майєро і Давид Делай ввели Field, тактику прийняття рішень для комутативних полів.

Крістін Полен змінила правила елімінації для порожніх і одиничних пропозіційних індуктивних типів.

Лоїк Поттьє розробив Fourier, тактику для розв'язання лінійних нерівностей на дійсних числах.

П'єр Крегут розробив нову, засновану на рефлексії версію процедури прийняття рішень Omega.

Клаудіо Сачердоті Коен розробив XML-вивід для модулів Coq для використання в Гіпертекстовій електронній бібліотеці математики (HELM cf http://www.cs.unibo.it/helm).

Бібліотека для ефективного представлення скінченних відображень за допомогою бінарних дерев, внесена Жаном Губо, була інтегрована в базові теорії.

П'єр Куртьє розробив команду і тактику для міркування про індуктивну структуру рекурсивно визначених функцій.

Яцек Хжонщ розробив і реалізував модульну систему Coq, основи якої лежать у докторській дисертації Жюдікаеля Курана.

Розробка координувалася К. Полен.

Багато дискусій у команді Démons і проєкті LogiCal значно вплинули на дизайн Coq, зокрема з Ж. Кураном, Ж. Дюпра, Ж. Губо, А. Мікелем, К. Марше, Б. Монате і Б. Вернером.

Інтенсивні користувачі пропонували покращення системи: І. Берто, Л. Поттьє, Л. Террі, П. Циммерманн з INRIA, К. Альварадо, П. Крегут, Ж.-Ф. Монін з France Telecom R & D.

Орсе, травень 2002 року
Х'юго Хербелін і Крістін Полен