Практикум по математической логике: Coq

Построение и проверка математических доказательств на компьютере

С. Л. Кузнецов

Московский государственный университет им. М.В. Ломоносова

Терминология

- Система автоматического поиска вывода (**prover**): по формуле ψ проверяет, доказуема ли ψ в данной системе (выдаёт доказательство или ответ «недоказуемо»).

Терминология

- Система автоматического поиска вывода (**prover**): по формуле ψ проверяет, доказуема ли ψ в данной системе (выдаёт доказательство или ответ «недоказуемо»).
    - SAT-солверы: “выполнима ли данная булева формула?”

Терминология

- Система автоматического поиска вывода (**prover**): по формуле ψ проверяет, доказуема ли ψ в данной системе (выдаёт доказательство или ответ «недоказуемо»).
  - SAT-солверы: “выполнима ли данная булева формула?”

    (ψ выполнима $\iff$ $\neg\psi$ *недоказуема* в классической логике)

Терминология

- Система автоматического поиска вывода (**prover**): по формуле ψ проверяет, доказуема ли ψ в данной системе (выдаёт доказательство или ответ «недоказуемо»).
  - SAT-солверы: “выполнима ли данная булева формула?”

    (ψ выполнима $\iff$ $\neg\psi$ *недоказуема* в классической логике)
  - логики описаний (модальные логики для представления баз знаний): OWL, биомедицинская информатика, ...

Терминология

- Система автоматического поиска вывода (**prover**): по формуле ψ проверяет, доказуема ли ψ в данной системе (выдаёт доказательство или ответ «недоказуемо»).
  - SAT-солверы: “выполнима ли данная булева формула?”
    (ψ выполнима $\iff$ $\neg\psi$ *недоказуема* в классической логике)
  - логики описаний (модальные логики для представления баз знаний): OWL, биомедицинская информатика, ...
  - разрешимые теории 1-го порядка: теория действительных чисел, элементарная геометрия, ...

Терминология

- Система автоматического поиска вывода (**prover**): по формуле ψ проверяет, доказуема ли ψ в данной системе (выдаёт доказательство или ответ «недоказуемо»).
  - SAT-солверы: “выполнима ли данная булева формула?”
    (ψ выполнима $\iff$ $\neg\psi$ *недоказуема* в классической логике)
  - логики описаний (модальные логики для представления баз знаний): OWL, биомедицинская информатика, ...
  - разрешимые теории 1-го порядка: теория действительных чисел, элементарная геометрия, ...
  - субструктурные логики для математической лингвистики: Grail, CatLog, ...

Терминология

- Система автоматического поиска вывода (**prover**): по формуле ψ проверяет, доказуема ли ψ в данной системе (выдаёт доказательство или ответ «недоказуемо»).
  - SAT-солверы: “выполнима ли данная булева формула?”
    (ψ выполнима $\iff$ $\neg\psi$ *недоказуема* в классической логике)
  - логики описаний (модальные логики для представления баз знаний): OWL, биомедицинская информатика, ...
  - разрешимые теории 1-го порядка: теория действительных чисел, элементарная геометрия, ...
  - субструктурные логики для математической лингвистики: Grail, CatLog, ...
  - ...

Терминология

- Система автоматического поиска вывода (**prover**): по формуле ψ проверяет, доказуема ли ψ в данной системе (выдаёт доказательство или ответ «недоказуемо»).
  - SAT-солверы: “выполнима ли данная булева формула?”
    (ψ выполнима $\iff$ $\neg\psi$ *недоказуема* в классической логике)
  - логики описаний (модальные логики для представления баз знаний): OWL, биомедицинская информатика, ...
  - разрешимые теории 1-го порядка: теория действительных чисел, элементарная геометрия, ...
  - субструктурные логики для математической лингвистики: Grail, CatLog, ...
  - ...

Интересные математические теории — арифметика, теория множеств, ... — неразрешимы, что делает prover невозможным.

Терминология

- Программа проверки доказательств (**proof checker**): принимает в качестве входных данных формулу A **и** некоторый объект (текст) $\mathscr{P}$ и возвращает «да», если $\mathscr{P}$ является доказательством формулы A (в данной логической системе), и «нет» в противном случае.

Терминология

- Программа проверки доказательств (**proof checker**): принимает в качестве входных данных формулу A **и** некоторый объект (текст) $\mathscr{P}$ и возвращает «да», если $\mathscr{P}$ является доказательством формулы A (в данной логической системе), и «нет» в противном случае.
    - Возможны практически для любых логических систем (с конечными доказательствами), по определению понятия «доказательство».

Терминология

- Программа проверки доказательств (**proof checker**): принимает в качестве входных данных формулу A **и** некоторый объект (текст) $\mathscr{P}$ и возвращает «да», если $\mathscr{P}$ является доказательством формулы A (в данной логической системе), и «нет» в противном случае.
  - Возможны практически для любых логических систем (с конечными доказательствами), по определению понятия «доказательство».
  - Однако полезность таких систем ограничена, поскольку работа по построению доказательства $\mathscr{P}$ полностью возлагается на человека.

Терминология

- Золотая середина — программы интерактивного поиска доказательства (**proof assistant**): интерактивное программное обеспечения для разработки формального доказательства.

Терминология

- Золотая середина — программы интерактивного поиска доказательства (**proof assistant**): интерактивное программное обеспечения для разработки формального доказательства.
    - Agda, Coq, HOL/Isabelle, Mizar, ...

Терминология

- Золотая середина — программы интерактивного поиска доказательства (**proof assistant**): интерактивное программное обеспечения для разработки формального доказательства.
    - Agda, Coq, HOL/Isabelle, Mizar, ...
- Доказательство, полученное в ходе интерактивной работы, потом проверяется программой проверки доказательств.

Типичное формальное доказательство

Докажем, что φ влечёт φ.

Типичное формальное доказательство

Докажем, что φ влечёт φ.

(1)	$(\varphi \to ((\varphi \to \varphi) \to \varphi) \to ((\varphi \to (\varphi \to \varphi)) \to (\varphi \to \varphi))$	акс. 2 с $A = C = \varphi$, $B = (\varphi \to \varphi)$
(2)	$\varphi \to ((\varphi \to \varphi) \to \varphi)$	акс. 1 с $A = \varphi$, $B = (\varphi \to \varphi)$
(3)	$(\varphi \to (\varphi \to \varphi)) \to (\varphi \to \varphi)$	MP (2) (1)
(4)	$\varphi \to (\varphi \to \varphi)$	акс. 1 с $A = B = \varphi$
(5)	$\varphi \to \varphi$	MP (4) (3)

Типичное формальное доказательство

Докажем, что φ влечёт φ.

(1)	$(\varphi \to ((\varphi \to \varphi) \to \varphi) \to ((\varphi \to (\varphi \to \varphi)) \to (\varphi \to \varphi))$	акс. 2 с $A = C = \varphi$, $B = (\varphi \to \varphi)$
(2)	$\varphi \to ((\varphi \to \varphi) \to \varphi)$	акс. 1 с $A = \varphi$, $B = (\varphi \to \varphi)$
(3)	$(\varphi \to (\varphi \to \varphi)) \to (\varphi \to \varphi)$	MP (2) (1)
(4)	$\varphi \to (\varphi \to \varphi)$	акс. 1 с $A = B = \varphi$
(5)	$\varphi \to \varphi$	MP (4) (3)

Бррр...

Формальные доказательства

- **Pro:** проверяются формально и автоматически, что гарантирует отсутствие «человеческого фактора» (особенно для сложных рассуждений).

Формальные доказательства

- **Pro:** проверяются формально и автоматически, что гарантирует отсутствие «человеческого фактора» (особенно для сложных рассуждений).
- **Contra:** сложны в построении и нечитаемы человеком — в случае неудачно выбранной аксиоматической системы!

Формальные доказательства

- **Pro:** проверяются формально и автоматически, что гарантирует отсутствие «человеческого фактора» (особенно для сложных рассуждений).
- **Contra:** сложны в построении и нечитаемы человеком — в случае неудачно выбранной аксиоматической системы!
- Даже в случае *удачной* аксиоматики разработка («программирование») формального доказательства — это отдельная работа!

Формальные доказательства

- **Pro:** проверяются формально и автоматически, что гарантирует отсутствие «человеческого фактора» (особенно для сложных рассуждений).
- **Contra:** сложны в построении и нечитаемы человеком — в случае неудачно выбранной аксиоматической системы!
- Даже в случае *удачной* аксиоматики разработка («программирование») формального доказательства — это отдельная работа!
- «Pro» перевешивают «contra» в случаях:

Формальные доказательства

- **Pro:** проверяются формально и автоматически, что гарантирует отсутствие «человеческого фактора» (особенно для сложных рассуждений).
- **Contra:** сложны в построении и нечитаемы человеком — в случае неудачно выбранной аксиоматической системы!
- Даже в случае *удачной* аксиоматики разработка («программирование») формального доказательства — это отдельная работа!
- «Pro» перевешивают «contra» в случаях:
    - когда доказательства получены компьютерным перебором;

Формальные доказательства

- **Pro:** проверяются формально и автоматически, что гарантирует отсутствие «человеческого фактора» (особенно для сложных рассуждений).
- **Contra:** сложны в построении и нечитаемы человеком — в случае неудачно выбранной аксиоматической системы!
- Даже в случае *удачной* аксиоматики разработка («программирование») формального доказательства — это отдельная работа!
- «Pro» перевешивают «contra» в случаях:
  - когда доказательства получены компьютерным перебором;
  - когда доказательство написано человеком, но в силу объёма и специфики не может быть разобрано *типичным* математиком;

Формальные доказательства

- **Pro:** проверяются формально и автоматически, что гарантирует отсутствие «человеческого фактора» (особенно для сложных рассуждений).
- **Contra:** сложны в построении и нечитаемы человеком — в случае неудачно выбранной аксиоматической системы!
- Даже в случае *удачной* аксиоматики разработка («программирование») формального доказательства — это отдельная работа!
- «Pro» перевешивают «contra» в случаях:
  - когда доказательства получены компьютерным перебором;
  - когда доказательство написано человеком, но в силу объёма и специфики не может быть разобрано *типичным* математиком;
  - верификации программного обеспечения.

Система Coq

Основные факты:

Система Coq

Основные факты:

- Разрабатывается в INRIA, Франция, с 1989 г.

Система Coq

Основные факты:

- Разрабатывается в INRIA, Франция, с 1989 г.
- Свободное программное обеспечение (LGPL 2.1), кросс-платформенное, доступно на сайте `https://coq.inria.fr/`

Система Coq

Основные факты:

- Разрабатывается в INRIA, Франция, с 1989 г.
- Свободное программное обеспечение (LGPL 2.1), кросс-платформенное, доступно на сайте `https://coq.inria.fr/`
- 2013 ACM Software System Award (T. Coquand et al.)

Система Coq

Основные факты:

- Разрабатывается в INRIA, Франция, с 1989 г.
- Свободное программное обеспечение (LGPL 2.1), кросс-платформенное, доступно на сайте `https://coq.inria.fr/`
- 2013 ACM Software System Award (T. Coquand et al.)
- Система написана на языке OCaml.

Система Coq

Основные факты:

- Разрабатывается в INRIA, Франция, с 1989 г.
- Свободное программное обеспечение (LGPL 2.1), кросс-платформенное, доступно на сайте `https://coq.inria.fr/`
- 2013 ACM Software System Award (T. Coquand et al.)
- Система написана на языке OCaml.
- Coq реализует **систему типов высших порядков**, которую также можно рассматривать как **функциональный язык программирования с зависимыми типами**

Система Coq

Основные факты:

- Разрабатывается в INRIA, Франция, с 1989 г.
- Свободное программное обеспечение (LGPL 2.1), кросс-платформенное, доступно на сайте `https://coq.inria.fr/`
- 2013 ACM Software System Award (T. Coquand et al.)
- Система написана на языке OCaml.
- Coq реализует **систему типов высших порядков**, которую также можно рассматривать как **функциональный язык программирования с зависимыми типами** (соответствие Карри – Говарда).
- Доказательства в coq не классические, а **интуиционистские (конструктивные).**

Применения Coq

- **Математические:** формализация сложных доказательств математических теорем

Применения Coq

- **Математические:** формализация сложных доказательств математических теорем
  - Знаменитый пример: **теорема о четырёх красках,** *любой планарный граф можно раскрасить в 4 цвета так, что смежные вершины разноцветны.*

Применения Coq

- **Математические:** формализация сложных доказательств математических теорем
  - Знаменитый пример: **теорема о четырёх красках,** *любой планарный граф можно раскрасить в 4 цвета так, что смежные вершины разноцветны.*
    Доказательство формализовал в Coq Г. Гонтье (Microsoft Research) в 2005 г.

Применения Coq

- **Математические:** формализация сложных доказательств математических теорем
  - Знаменитый пример: **теорема о четырёх красках,** *любой планарный граф можно раскрасить в 4 цвета так, что смежные вершины разноцветны.*
    Доказательство формализовал в Coq Г. Гонтье (Microsoft Research) в 2005 г.
  - Другой пример: теорема Фейта–Томпсона о группах нечётного порядка.

Применения Coq

- **Математические:** формализация сложных доказательств математических теорем
  - Знаменитый пример: **теорема о четырёх красках,** *любой планарный граф можно раскрасить в 4 цвета так, что смежные вершины разноцветны.*
    Доказательство формализовал в Coq Г. Гонтье (Microsoft Research) в 2005 г.
  - Другой пример: теорема Фейта–Томпсона о группах нечётного порядка.
    Формализовал Г. Гонтье в 2012 г.

Применения Coq

- **Программирование:** *верификация* программного обеспечения, т.е. формальное доказательство его соответствия формально заданной спецификации

Применения Coq

- **Программирование:** *верификация* программного обеспечения, т.е. формальное доказательство его соответствия формально заданной спецификации
  ... и даже *извлечение* работающего и верифицированного кода из интуиционистских доказательств!

Применения Coq

- **Программирование:** *верификация* программного обеспечения, т.е. формальное доказательство его соответствия формально заданной спецификации
  ... и даже *извлечение* работающего и верифицированного кода из интуиционистских доказательств!
  - Пример: CompCert, сертифицированный компилятор C (К. Леруа, 2008).

Применения Coq

- **Программирование:** *верификация* программного обеспечения, т.е. формальное доказательство его соответствия формально заданной спецификации
  ... и даже *извлечение* работающего и верифицированного кода из интуиционистских доказательств!
  - Пример: CompCert, сертифицированный компилятор C (К. Леруа, 2008).
  - В целом, верификация гарантирует большую надёжность, чем тестирование (но всё равно не абсолютную).

Особенности Coq

- Логика Coq **интуиционистская,** т.е. без закона исключённого третьего (его можно при желании добавить как дополнительную аксиому).

Особенности Coq

- Логика Coq **интуиционистская,** т.е. без закона исключённого третьего (его можно при желании добавить как дополнительную аксиому).
- Логика **высшего порядка:** можно навешивать кванторы по чему угодно.

Особенности Coq

- Логика Coq **интуиционистская,** т.е. без закона исключённого третьего (его можно при желании добавить как дополнительную аксиому).
- Логика **высшего порядка:** можно навешивать кванторы по чему угодно.
- Исчисление **естественного вывода;** правила вывода реализованы в виде **тактик.**

Особенности Coq

- Логика Coq **интуиционистская,** т.е. без закона исключённого третьего (его можно при желании добавить как дополнительную аксиому).
- Логика **высшего порядка:** можно навешивать кванторы по чему угодно.
- Исчисление **естественного вывода;** правила вывода реализованы в виде **тактик.**
- Соответствие Карри – Говарда: формулы как типы данных, доказательства как программы.

Конструктивные и неконструктивные доказательства

Утверждение: существуют такие иррациональные числа α и β, что α^β рационально.

Конструктивные и неконструктивные доказательства

Утверждение: существуют такие иррациональные числа α и β, что α^β рационально.

- Конструктивное доказательство: $\alpha = \sqrt{2}$, $\beta = \log_{\sqrt{2}} 3$.

Конструктивные и неконструктивные доказательства

Утверждение: существуют такие иррациональные числа α и β, что α^β рационально.

- Конструктивное доказательство: $\alpha = \sqrt{2}$, $\beta = \log_{\sqrt{2}} 3$.
- Неконструктивное доказательство: рассмотрим два случая:

  1. $\sqrt{2}^{\sqrt{2}}$ рационально. Тогда возьмём $\alpha = \beta = \sqrt{2}$.
  2. $\sqrt{2}^{\sqrt{2}}$ иррационально. Тогда возьмём $\alpha = \sqrt{2}^{\sqrt{2}}$ и $\beta = \sqrt{2}$ ($\alpha^\beta = 2$ рационально).

Интуиционистская логика

- Интуиционистская логика получается из классической исключением аксиомы $A \vee \neg A$ (закона исключённого третьего).

Интуиционистская логика

- Интуиционистская логика получается из классической исключением аксиомы $A \vee \neg A$ (закона исключённого третьего).
- В логике без этого закона неконструктивные доказательства невозможны.

Интуиционистская логика

- Интуиционистская логика получается из классической исключением аксиомы $A \vee \neg A$ (закона исключённого третьего).
- В логике без этого закона неконструктивные доказательства невозможны.
- Если в Coq нужно записать неклассическое доказательство, закон исключённого третьего добавляют как дополнительную аксиому (“`classic`”).

Тактики и естественный вывод

- Работа с Coq’ом интерактивна.

Тактики и естественный вывод

- Работа с Coq'ом интерактивна.
- В каждый момент имеется несколько *задач на доказательство* вида $x_1 : A_1, \ldots, x_n : A_n \vdash B$.

Тактики и естественный вывод

- Работа с Coq'ом интерактивна.
- В каждый момент имеется несколько *задач на доказательство* вида $x_1 : A_1, \ldots, x_n : A_n \vdash B$.
- A_i — посылки, B — требуемое заключение.

Тактики и естественный вывод

- Работа с Coq'ом интерактивна.
- В каждый момент имеется несколько *задач на доказательство* вида $x_1 : A_1, \ldots, x_n : A_n \vdash B$.
- A_i — посылки, B — требуемое заключение.
- Если $B = A_i$, доказывать ничего не надо.

Тактики и естественный вывод

- Работа с Coq'ом интерактивна.
- В каждый момент имеется несколько *задач на доказательство* вида $x_1 : A_1, \ldots, x_n : A_n \vdash B$.
- A_i — посылки, B — требуемое заключение.
- Если $B = A_i$, доказывать ничего не надо.
- Применение тактики призвано *упростить* задачу поиска вывода.

Тактики и естественный вывод

- Работа с Coq'ом интерактивна.
- В каждый момент имеется несколько *задач на доказательство* вида $x_1 : A_1, \ldots, x_n : A_n \vdash B$.
- A_i — посылки, B — требуемое заключение.
- Если $B = A_i$, доказывать ничего не надо.
- Применение тактики призвано *упростить* задачу поиска вывода.
- Например, от задачи $\Gamma \vdash (B \longrightarrow C)$ можно перейти к $\Gamma, B \vdash C$, а от задачи $\Gamma \vdash B_1 \vee B_2$ — к одной из задач $\Gamma \vdash B_i$ ($i = 1$ или 2).

Тактики и естественный вывод

- Работа с Coq'ом интерактивна.
- В каждый момент имеется несколько *задач на доказательство* вида $x_1 : A_1, \ldots, x_n : A_n \vdash B$.
- A_i — посылки, B — требуемое заключение.
- Если $B = A_i$, доказывать ничего не надо.
- Применение тактики призвано *упростить* задачу поиска вывода.
- Например, от задачи $\Gamma \vdash (B \longrightarrow C)$ можно перейти к $\Gamma, B \vdash C$, а от задачи $\Gamma \vdash B_1 \vee B_2$ — к одной из задач $\Gamma \vdash B_i$ ($i = 1$ или 2).
- Простейшие тактики — это правила вывода CIC.

Тактика assumption

Тактика intro

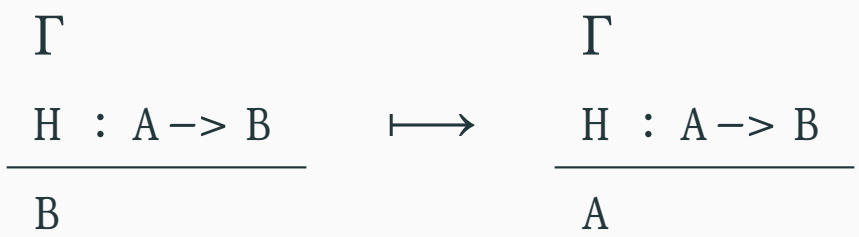
$$\frac{\Gamma}{\texttt{forall x:T, A(x)}} \quad \longmapsto \quad \frac{\Gamma}{\texttt{x : T}} \qquad \texttt{A(x)}$$

Тактика intro

$$\frac{\Gamma}{\texttt{forall x:T, A(x)}} \quad \longmapsto \quad \frac{\Gamma}{\texttt{x : T}}$$

$$\frac{\Gamma}{\texttt{A -> B}} \quad \longmapsto \quad \frac{\Gamma}{\texttt{H : A}}$$

Тактика apply



Тактика apply

$$\frac{\Gamma \quad H : A \to B}{B} \longmapsto \frac{\Gamma \quad H : A \to B}{A}$$

$$\frac{\Gamma \quad H : A \to B \to C}{C} \longmapsto \frac{\Gamma \quad H : A \to B \to C}{A}, \quad \frac{\Gamma \quad H : A \to B \to C}{B}$$

Конъюнкция

```
Inductive and (A B : Prop) : Prop :=
conj : A -> B -> A ∧ B .
```

Конъюнкция

```
Inductive and (A B : Prop) : Prop :=
conj : A -> B -> A ∧ B .
```

Тактика `split` (`apply conj`):

$$\frac{\Gamma}{A \wedge B} \quad \longmapsto \quad \frac{\Gamma}{A}\,, \quad \frac{\Gamma}{B}\,.$$

Конъюнкция

```
Inductive and (A B : Prop) : Prop :=
conj : A -> B -> A ∧ B .
```

Тактика `split` (`apply conj`):

$$\frac{\Gamma}{A \wedge B} \quad \longmapsto \quad \frac{\Gamma}{A}, \quad \frac{\Gamma}{B}.$$

Тактика `elim`.

$$\frac{\begin{array}{l}\Gamma \\ H : A \wedge B\end{array}}{C} \quad \longmapsto \quad \frac{\Gamma}{A \to B \to C}.$$

Дизъюнкция

```
Inductive or (A B : Prop) : Prop :=
or_introl : A -> A ∨ B
| or_intror : B -> A ∨ B .
```

Тактики `left` и `right`:

$$\frac{\Gamma}{A \vee B} \longmapsto \frac{\Gamma}{A} \quad \text{и} \quad \frac{\Gamma}{A \vee B} \longmapsto \frac{\Gamma}{B}$$

Тактика `elim`:

$$\frac{\Gamma}{\texttt{H : A} \vee \texttt{B}} \longmapsto \frac{\Gamma}{\texttt{H : A} \vee \texttt{B}}, \frac{\Gamma}{\texttt{H : A} \vee \texttt{B}}$$
$$\texttt{C} \qquad\qquad \texttt{A -> C} \qquad\qquad \texttt{B -> C}$$

Константы True и False, отрицание

```
Inductive False : Prop := .
```

Константы True и False, отрицание

```
Inductive False : Prop := .

Inductive True : Prop := I : True .
```

Константы True и False, отрицание

```
Inductive False : Prop := .

Inductive True : Prop := I : True .

not = fun A : Prop => A -> False : Prop -> Prop
```

Константы True и False, отрицание

```
Inductive False : Prop := .

Inductive True : Prop := I : True .

not = fun A : Prop => A -> False : Prop -> Prop
```

Раскрытие отрицания — тактика `red` (преобразование по определению в импликацию ко лжи).

Тактика contradict

$$\frac{\Gamma \quad H : \sim A}{B} \longmapsto \frac{\Gamma}{A} \qquad \frac{\Gamma \quad H : A}{B} \longmapsto \frac{\Gamma}{\sim A}$$

$$\frac{\Gamma \quad H : \sim A}{\sim B} \longmapsto \frac{\Gamma \quad H : B}{A} \qquad \frac{\Gamma \quad H : A}{\sim B} \longmapsto \frac{\Gamma \quad H : B}{\sim A}$$

Классическая логика

Закон исключённого третьего:

```
Axiom classic: forall P : Prop, P \/ ~P.
```

Классическая логика

Закон исключённого третьего:

```
Axiom classic: forall P : Prop, P \/ ~P.
```

```
Theorem NNPP: forall p : Prop, ~~p -> p.
Theorem proof_irrelevance: forall (P : Prop)(p1 p2 : P), p1 =
p2.
```

Классическая логика

Закон исключённого третьего:

```
Axiom classic: forall P : Prop, P \/ ~P.
```

```
Theorem NNPP: forall p : Prop, ~~p -> p.
Theorem proof_irrelevance: forall (P : Prop)(p1 p2 : P), p1 =
p2.
```

Модуль: `Require Import Classical.`

(Интуиционистский) квантор существования

```
Inductive ex (A : Type) (P : A -> Prop) : Prop :=

   ex_intro : forall x : A, P x -> ex P.
```

(Интуиционистский) квантор существования

```
Inductive ex (A : Type) (P : A -> Prop) : Prop :=

  ex_intro : forall x : A, P x -> ex P.
```

Команда “`exists a.`”

$$\frac{\Gamma}{\texttt{exists x : A,P x}} \quad \longmapsto \quad \frac{\Gamma}{\texttt{P a}}$$

(Интуиционистский) квантор существования

```
Inductive ex (A : Type) (P : A -> Prop) : Prop :=

  ex_intro : forall x : A, P x -> ex P.
```

Команда “`exists a.`”

$$\frac{\Gamma}{\texttt{exists x : A,P x}} \quad \longmapsto \quad \frac{\Gamma}{\texttt{P a}}$$

Команда “`elim H.`”

$$\frac{\begin{array}{l}\Gamma \\ \texttt{H : exists x : A,P x}\end{array}}{\texttt{B}} \quad \longmapsto \quad \frac{\begin{array}{l}\Gamma \\ \texttt{H : exists x : A,P x}\end{array}}{\texttt{forall x : A, P x -> B}}$$

Сечение

Правило сечения

$$\frac{\Gamma \vdash A \quad \Gamma, A \vdash B}{\Gamma \vdash B}$$

реализовано в виде тактики `assert`, которой в качестве параметра надо передавать высекаемую формулу (команда “`assert A.`”).

$$\frac{\Gamma}{\texttt{B}} \quad \longmapsto \quad \frac{\Gamma}{\texttt{A}} \quad , \quad \frac{\begin{array}{c}\Gamma \\ \texttt{H : A}\end{array}}{\texttt{B}} \quad .$$

Сечение

Правило сечения

$$\frac{\Gamma \vdash A \quad \Gamma, A \vdash B}{\Gamma \vdash B}$$

реализовано в виде тактики `assert`, которой в качестве параметра надо передавать высекаемую формулу (команда “`assert A.`”).

$$\frac{\Gamma}{\texttt{B}} \quad \longmapsto \quad \frac{\Gamma}{\texttt{A}} \quad , \quad \frac{\begin{array}{c}\Gamma \\ \texttt{H : A}\end{array}}{\texttt{B}} \; .$$

Близкий эффект достигается также командой “`cut A.`”, реализующей правило Modus Ponens:

$$\frac{\Gamma}{\texttt{B}} \quad \longmapsto \quad \frac{\Gamma}{\texttt{A -> B}} \quad , \quad \frac{\Gamma}{\texttt{A}} \; .$$