Logika i teoria typów

Wykład 4

9 listopada 2021

Tydzień temu: algorytm Ben-Yellesa

To answer $\Gamma \vdash ? : \alpha$, apply one of the following tactics:

- For $\alpha = \beta \to \gamma$, ask $\Gamma \cup \{x : \beta\} \vdash ? : \gamma$ (fresh x).

  (Solution $M = \lambda x{:}\beta.\, N^{\gamma}$.)

- Find $x : \beta_1 \to \cdots \to \beta_k \to \alpha$ in Γ,
  then ask $\Gamma \vdash ? : \beta_i$, for all i. Success if $k = 0$.

  (Solution $M = xN_1^{\beta_1} \ldots N_k^{\beta_k}$.)

W drugim przypadku można żądać, aby α była zmienną.

Postaci normalne (fragment implikacyjny)

Konstruktory: $\lambda x : \alpha.\, N$

Eliminatory: $xN_1 \ldots N_k$

Poszukiwanie dowodu normalnego:

To prove τ, use an assumption with suffix τ.
Otherwise proof is a constructor.

Dowody normalne (definicja I)

Strategie ∃rosa w przypadku ogólnym:

- zmienne;
- $M[y := xN]$;
- $M[y := x\{i\}]$, gdzie $i = 1, 2$;
- case x of $[y]\, M$ or $[z]\, N$;
- $\lambda x : \beta.\, M$;
- $\langle M, N \rangle$;
- $\mathsf{in}_i(M)$;
- $\varepsilon_\tau(x)$.

Dowody normalne (definicja II)

- *Introductions:*
  – $\lambda x : \alpha.\, N^{\beta} : \alpha \to \beta$,
  – $\langle N_1^{\alpha}, N_2^{\beta} \rangle : \alpha \wedge \beta$,
  – $\mathsf{in}_i(N^{\alpha}) : \alpha \vee \beta$;
- *Proper eliminators (P):*
  – $x^{\alpha} : \alpha$,
  – $P^{\alpha \to \beta} N^{\alpha} : \beta$,
  – $P^{\alpha_1 \wedge \alpha_2}\{i\} : \alpha_i$;
- *Improper eliminators:*
  $\varepsilon_\varphi(P^{\perp}) : \varphi$,
  case $P^{\alpha \vee \gamma}$ of $[u^{\alpha}]M^{\beta}$ or $[v^{\gamma}]N^{\beta} : \beta$

Poprawność (do domu)

Te dwie definicje są równoważne.
Dowód wykorzystuje taki techniczny lemat:

Lemat: *Jeśli M jest dowodem normalnym (w sensie definicji II), a P jest eliminatorem właściwym, to także $M[x := P]$ jest dowodem normalnym.*

Pożytek z normalizacji: niesprzeczność

Wniosek: *Intuicjonistyczny rachunek zdań jest niesprzeczny: formuła $\perp$ nie ma dowodu.*

Dowód: Wtedy istniałby dowód normalny, czyli taki term M w postaci normalnej, że $\varnothing \vdash M : \perp$. Jak on może wyglądać?

- Nie może być konstruktorem, bo nie ma konstruktora dla $\perp$ (nie ma reguły $(W\perp)$);
- Nie może być właściwym eliminatorem postaci $xE_1 \ldots E_m$, bo nie ma zmiennych.
- Zatem nie ma też eliminatorów postaci $\varepsilon_\perp(P)$, ani case $P \ldots$, bo do tego potrzebny właściwy eliminator P.

Dowody normalne (definicja III?)

To powinny być postaci normalne
ze względu na odpowiednie redukcje.
Beta-redukcje?

$(\to)$ $(\lambda x^{\tau}\, M^{\sigma})N^{\tau} \implies M[x := N] : \sigma$.

$(\wedge)$ $\langle M^{\tau}, N^{\sigma} \rangle\{1\} \implies M : \tau, \qquad \langle M^{\tau}, N^{\sigma} \rangle\{2\} \implies N : \sigma$.

$(\vee)$ **case in$_1$** (P^{τ}) **of** $[x^{\tau}]M^{\rho}, [y^{\sigma}]N^{\rho} \implies M[x := P] : \rho$
case in$_2$ (Q^{σ}) **of** $[x^{\tau}]M^{\rho}, [y^{\sigma}]N^{\rho} \implies N[y := Q] : \rho$.

Beta-redukcje to za mało

Term w postaci beta-normalnej może np. wyglądać tak:

$$(\textbf{case } z^{\tau\vee\sigma} \textbf{ of } [x^{\tau}]P^{\alpha\to\beta}, [y^{\sigma}]Q^{\alpha\to\beta})M^{\alpha}$$

Jak nad tym zapanować?

Wprowadzić dodatkowe redukcje porządkujące (*permutacje*).

Permutacje

Kłopotliwa sytuacja:

Brzydka eliminacja, a potem znowu eliminacja, czyli:

$(\textbf{case } N \textbf{ of } [x]P \textbf{ or } [y]Q)E \qquad (\varepsilon_{\sigma}(M))E$

Eliminator E nie ma dostępu do „wnętrza" termu.

Permutacje:

$(\textbf{case } N \textbf{ of } [x]P \textbf{ or } [y]Q)E \;\Rightarrow\; \textbf{case } N \textbf{ of } [x]PE \textbf{ or } [y]QE$

$(\varepsilon_{\sigma}(M^{\perp}))E^{\tau} \;\Rightarrow\; \varepsilon_{\tau}(M^{\perp})$

Permutacje dla $\perp$

- $\varepsilon_{\psi}(\varepsilon_{\perp}(M)) \;\Rightarrow\; \varepsilon_{\psi}(M)$;
- $\varepsilon_{\varphi\to\psi}(M)N \;\Rightarrow\; \varepsilon_{\psi}(M)$;
- $\varepsilon_{\varphi_1\wedge\varphi_2}(M)\{i\} \;\Rightarrow\; \varepsilon_{\varphi_i}(M)$;
- $\textbf{case } \varepsilon_{\sigma\vee\tau}(M) \textbf{ of } [u]R^{\rho} \textbf{ or } [v]S^{\rho} \;\Rightarrow\; \varepsilon_{\rho}(M)$.

Schemat:

$\varepsilon_{\tau}(M)E^{\sigma} \Rightarrow \varepsilon_{\sigma}(M)$, czyli $M[\tau]E^{\sigma} \Rightarrow M[\sigma]$

Permutacje dla **case**

- $\varepsilon_{\varphi}(\texttt{case } M \texttt{ of } [x]P \texttt{ or } [y]Q) \;\Rightarrow\; \texttt{case } M \texttt{ of } [x]\varepsilon_{\varphi}(P) \texttt{ or } [y]\varepsilon_{\varphi}(Q)$;
- $(\texttt{case } M \texttt{ of } [x]P \texttt{ or } [y]Q)N \;\Rightarrow\; \texttt{case } M \texttt{ of } [x]PN \texttt{ or } [y]QN$;
- $(\texttt{case } M \texttt{ of } [x]P \texttt{ or } [y]Q)\{i\} \;\Rightarrow\; \texttt{case } M \texttt{ of } [x]P\{i\} \texttt{ or } [y]Q\{i\}$;
- $\texttt{case } (\texttt{case } M \texttt{ of } [x]P \texttt{ or } [y]Q) \texttt{ of } [u]R \texttt{ or } [v]S \;\Rightarrow\; \texttt{case } M \texttt{ of } [x](\texttt{case } P \texttt{ of } [u]R \texttt{ or } [v]S) \texttt{ or } [y](\texttt{case } Q \texttt{ of } [u]R \texttt{ or } [v]S)$

Schemat: $M[x.P^{\tau}, y.Q^{\tau}]E^{\sigma} \Rightarrow M[x.P^{\tau}E^{\sigma}, y.Q^{\tau}E^{\sigma}]$

Dobra wiadomość

Postaci normalne ze względu na beta i permutacje,
to dokładnie dowody normalne:

Konstruktory: $\lambda x:\alpha.\, N$, $\langle N_1, N_2\rangle$, $\mathrm{in}_i(N)$;

Ładne eliminatory: x, PN, $P\{i\}$;

Brzydkie eliminatory: $\varepsilon_{\varphi}(P)$, case P of $[x]N_1$ or $[y]N_2$.

Eliminatory właściwe

- *Proper eliminators (P):*
  - $x^{\alpha} : \alpha$,
  - $P^{\alpha\to\beta}N^{\alpha} : \beta$,
  - $P^{\alpha_1\wedge\alpha_2}\{i\} : \alpha_i$;

Taki eliminator właściwy ma zawsze postać $xE_1E_2\ldots E_n$, gdzie E_i to albo termy, albo rzutowania $\{1\}$, $\{2\}$.

Typ eliminatora właściwego jest "końcówką" typu zmiennej.

Przykład niewłaściwej eliminacji

W otoczeniu:

$x : (q\wedge p)\vee r$, $y : r\to(r\to p)\vee q$, $z : q\to p$

term

case x of $[u]u\{2\}$ or $[v]$ (case yv of $[w_1]w_1v$ or $[w_2]zw_2$)

ma typ p.

Typy zmiennych: $u : q\wedge p$, $v : r$, $w_1 : r\to p$, $w_2 : q$.

Example application:

Disjunction property: *If $\Gamma\vdash\alpha\vee\beta$, and $\vee$ does not occur in Γ, then $\Gamma\vdash\alpha$ or $\Gamma\vdash\beta$.*

Proof: Show that if $\Gamma\vdash M:\alpha\vee\beta$, and M is normal, then either $\Gamma\vdash\perp$ or M is a constructor **in**.

Types of proper eliminators contain no $\vee$, so M is either an introduction (OK) or an „improper" eliminator.

If $M=\varepsilon(M')$ then $\Gamma\vdash M':\perp$ (OK).

If $M=$ case $M'\ldots$ then M' is a proper eliminator of a disjunction type – excluded as above.

Wnioski:

1. Prawo wyłączonego środka $p \vee \neg p$ nie ma dowodu.
2. Prawo De Morgana $\neg(p \wedge q) \to \neg p \vee \neg q$ też nie ma.

Separation theorem (Mordchaj Wajsberg, 1938)

Twierdzenie: *Spójniki intuicjonistyczne nie są wzajemnie definiowalne.*

Disjunction: *There is no* α *without* $\vee$ *such that* $p \vee q \leftrightarrow \alpha$.

Proof:

If $\alpha \vdash p \vee q$ then $\alpha \vdash p$ or $\alpha \vdash q$. But $p \vee q \nvdash p$ and $p \vee q \nvdash q$.

Separation theorem

Implication: *There is no* α *without* $\to$ *such that* $p \to q \vdash \alpha$.

Proof: Induction.

If $\alpha = \beta \wedge \gamma$ then $p \to q \vdash \beta$.

If $\alpha = \beta \vee \gamma$ then either $p \to q \vdash \beta$ or $p \to q \vdash \gamma$.

Separation theorem

Conjunction: *If* $\alpha \vdash p \wedge q$, *and* $\wedge$ *is not in* α, *then* $\alpha \vdash \bot$.

Proof: If $\alpha = \beta \vee \gamma$ then $\beta \vdash \bot$ and $\gamma \vdash \bot$ by induction.

Assume that $\alpha = \beta \to \gamma$.

Then also $\beta \to \bot \vdash p$,
whence $\beta \to \bot \vdash \bot$, as p is not an available target.

On the other hand, also $\gamma \vdash p \wedge q$, so $\gamma \vdash \bot$, by induction.

Thus we have $\beta \to \gamma \vdash \beta \to \bot$,
and therefore $\beta \to \gamma \vdash \bot$.

Nagroda pocieszenia

Strzałka Kuzniecowa:
$K(p, q, r) = ((p \vee q) \wedge \neg r) \vee (\neg p \wedge (q \leftrightarrow r))$.

Fakt: Spójniki $\bot$, $\to$, $\vee$, $\wedge$ są definiowalne z pomocą strzałki Kuzniecowa.

Uwaga: Ale tu nie ma nic takiego, jak funkcyjna zupełność logiki klasycznej.

Poszukiwanie dowodu normalnego:

Dowód formuły τ może być:

- Konstruktorem (o ile τ nie jest atomem);
- Eliminatorem o zmiennej czołowej typu σ:
  - Właściwym, gdy τ jest „sufiksem" typu σ;
  - Niewłaściwym, gdy σ ma „sufiks" postaci $\alpha \vee \beta$ lub $\bot$.

Drobne usprawnienie: Można stosować drugi przypadek tylko wtedy, gdy τ jest atomem lub alternatywą.

To jest *algorytm Wajsberga-Bezaniszwilego*.

Algorytm Wajsberga (uzupełniony)

To answer $\Gamma \vdash \alpha$?, apply one of the following tactics:

- For $\alpha = \beta \to \gamma$, ask if $\Gamma, \beta \vdash \gamma$?
- For $\alpha = \beta \wedge \gamma$, ask if $\Gamma \vdash \beta$ AND $\Gamma \vdash \gamma$.
- For $\alpha = \alpha_1 \vee \alpha_2$, GUESS $i \in \{1, 2\}$, and ask if $\Gamma \vdash \alpha_i$.
- GUESS $\beta_1 \to \cdots \to \beta_k \to A$ from Γ, where $A = \alpha$ or $A = \bot$; then ask FOR ALL i, if $\Gamma \vdash \beta_i$. (Sukces, gdy $k = 0$.)
- GUESS $\beta_1 \to \cdots \to \beta_k \to \beta \vee \gamma$ from Γ, and ask FOR ALL i, if $\Gamma \vdash \beta_i$. Also ask if $\Gamma, \beta \vdash \alpha$ AND $\Gamma, \gamma \vdash \alpha$.
- GUESS $\beta_1 \to \cdots \to \beta_k \to \beta \wedge \gamma$ from Γ, and ask FOR ALL i, if $\Gamma \vdash \beta_i$. Also ask if $\Gamma, \beta, \gamma \vdash \alpha$.

Alternacja

Algorytm Wajsberga rozgałęzia się:

- Egzystencjalnie (niedeterministycznie), bo można wybrać różne taktyki i różne zmienne;
- Uniwersalnie (rekurencyjnie), bo trzeba niezależnie rozwiązywać różne zadania.

To się nazywa *alternacja*.

Terminacja

W każdej gałęzi obliczenia:

- Używa się tylko podformuł zadania początkowego.
- Otoczenie Γ nigdy nie maleje.
- Można ignorować powtarzające się założenia (utożsamiać zmienne tego samego typu).

Morał: Po wielomianowej liczbie kroków zadanie musi się powtórzyć. Złożoność tego algorytmu jest wielomianowa w czasie alternującym.

Zatem problem decyzyjny dla intuicjonistycznego rachunku zdań jest w klasie Pspace.

Algorytm Wajsberga: operacje

Zadanie ma postać $\Gamma \vdash \alpha$.
Jakie operacje na nim wykonujemy?

- GUESS $\beta_1 \to \cdots \to \beta_k \to A$ from Γ, where $A = \alpha$ or $A = \bot$; then ask if $\Gamma \vdash \beta_i$, FOR ALL i.
  Rozgałęzienie rekurencyjne (uniwersalne)e ze zmianą celów. Wymaga dostępności odp. założenia.
- GUESS $\beta_1 \to \cdots \to \beta_k \to \beta \vee \gamma$ from Γ, and ask if $\Gamma \vdash \beta_i$, FOR ALL i. Also ask if $\Gamma, \beta \vdash \alpha$ AND $\Gamma, \gamma \vdash \alpha$.
  Rozgałęzienie uniwersalne ze zmianą celów. W niektórych przypadkach dodanie założenia.
- GUESS $\beta_1 \to \cdots \to \beta_k \to \beta \wedge \gamma$ from Γ, and ask FOR ALL i, if $\Gamma \vdash \beta_i$. Also ask if $\Gamma, \beta, \gamma \vdash \alpha$.
  Podobnie.

Algorytm Wajsberga: operacje

Zadanie ma postać $\Gamma \vdash \alpha$.
Jakie operacje na nim wykonujemy?

- For $\alpha = \beta \to \gamma$, ask if $\Gamma, \beta \vdash \gamma$?
  Zmiana celu plus dodanie nowego założenia. Wymaga celu odpowiedniej postaci.
- For $\alpha = \beta \wedge \gamma$, ask if $\Gamma \vdash \beta$ AND $\Gamma \vdash \gamma$.
  Rekurencyjne wywołanie dla dwóch nowych celów (rozgałęzienie uniwersalne). Wymaga odp. celu.
- For $\alpha = \alpha_1 \vee \alpha_2$, GUESS $i \in \{1, 2\}$, and ask if $\Gamma \vdash \alpha_i$.
  Egzystencjalny wybór ze zmianą celu.

Algorytm Wajsberga: operacje

Zadanie ma postać $\Gamma \vdash \alpha$.
Jakie operacje na nim wykonujemy?

- Dodanie założenia;
- Zmiana celu;
- Sprawdzenie dostępności założenia;
- Sprawdzenie postaci celu;
- Rozgałęzienie egzystencjalne;
- Rozgałęzienie uniwersalne.

Algorytm Wajsberga jako automat

- Cel dowodowy to stan maszyny;
- Otoczenie Γ to zawartość pamięci.
- Operacje:
  - Dodanie założenia = podniesienie flagi;
  - Zmiana celu = zmiana stanu;
  - Sprawdzenie dostępności założenia = sprawdzenie flagi;
  - Sprawdzenie postaci celu = sprawdzenie stanu maszyny;
  - Rozgałęzienie egzystencjalne;
  - Rozgałęzienie uniwersalne.

Uwaga: (1) nie można opuścić flagi (usunąć danej z pamięci); (2) nie można sprawdzić, że flaga jest opuszczona.

Monotonic automata

Monotonic automaton: $\mathcal{M} = \langle Q, R, f, \mathcal{I} \rangle$,

- Q is a finite set of states, $f \in Q$ the final state.
- R is a finite set of registers;
- $\mathcal{I}$ is a finite set of instructions of the form:
  (1) q : `check` S_1; `set` S_2; `jmp` p,
  (2) q : `jmp` p_1 `and` p_2,
  where $p, p_1, p_2 \in Q$ and $S_1, S_2 \subseteq R$.

Monotonic automata

Configuration: $\langle q, S \rangle$, where $q \in Q$ and $S \subseteq R$.

Transitions:

- for $I = q$: `check` S_1; `set` S_2; `jmp` p:
  $\langle q, S \rangle \to_I \langle p, S \cup S_2 \rangle$, provided $S_1 \subseteq S$;
  If $\langle p, S \cup S_2 \rangle$ accepting then $\langle q, S \rangle$ accepting
- for $I = q$: `jmp` p_1 `and` p_2:
  $\langle q, S \rangle \to_I \langle p_1, S \rangle$ **and** $\langle q, S \rangle \to_I \langle p_2, S \rangle$
  If $\langle p_1, S \rangle$ and $\langle p_2, S \rangle$ accepting then so is $\langle q, S \rangle$.

Proofs into programs

Twierdzenie: *Problem dowodzenia w intuicjonistycznym rachunku zdań sprowadza się (w pamięci logarytmicznej) do problemu stopu dla automatów monotonicznych.*

Dowód: Dla danej formuły φ konstruujemy automat:

- Stany = możliwe cele: podformuły formuły φ i jeden stan końcowy;
- Rejestry = możliwe założenia, czyli też podformuły.
- Instrukcje = jak w algorytmie Wajsberga.

Trzeba jeszcze dodać przejścia do stanu końcowego z konfiguracji wygrywających ∃rosa.

Monotonic automata

Lemma

The halting problem for monotonic automata

Is a given configuration accepting?

is PSPACE*-hard.*

Cześć łatwa: Problem stopu rozstrzygamy w pamięci wielomianowej, bo zbiór rejestrów tylko rośnie.

Część trudna: Monotonic encoding of an ATM

Alternating time $p(n)$.

Registers: $stan_s^t$, lit_{ia}^t, poz_i^t,

Initial configuration:

$C_0 = \langle loop_1^0, \{lit_{1a_1}^0, ..., lit_{na_n}^0, lit_{n+1B}^0, ..., lit_{p(n)B}^0, stan_0^0, poz_1^0\}\rangle$.

represents initial ID of TM for input $w = a_1...a_n$.

Later:

$C = \langle loop_1^t, S\rangle$ encodes the first t steps of TM computation,

e.g., $lit_{6a}^5 \in S$ iff after 5 steps, the 6th cell contains a.

Example instructions for $(s, a) \Rightarrow (u, b, +\varepsilon)$

When $i < p(n)$:

$$loop_i^t :$$
$$\texttt{check } stan_s^t, poz_i^t, lit_{ia}^t; \texttt{set } stan_u^{t+1}, poz_{i+\varepsilon}^{t+1}, lit_{ib}^{t+1}; \texttt{jmp } loop_{i+1}^t;$$
$$\texttt{check } stan_s^t, poz_j^t, lit_{ia}^t; \texttt{set } lit_{ia}^{t+1}; \texttt{jmp } loop_{i+1}^t;$$

When $i = p(n)$:

$$loop_i^t :$$
$$\texttt{check } stan_s^t, poz_i^t, lit_{ia}^t; \texttt{set } stan_u^{t+1}, poz_{i+\varepsilon}^{t+1}, lit_{ib}^{t+1}; \texttt{jmp } loop_1^{t+1};$$
$$\texttt{check } stan_s^t, poz_j^t, lit_{ia}^t; \texttt{set } lit_{ia}^{t+1}; \texttt{jmp } loop_1^{t+1}.$$

Example instructions: universal split

Assume that TM does not write nor move in universal states.

When $i < p(n)$:

$loop_i^t : \texttt{check } stan_s^t, lit_{ia}^t, poz_j^t; \texttt{set } lit_{ia}^{t+1}, poz_j^{t+1}; \texttt{jmp } loop_{i+1}^t;$

When $i = p(n)$:

$loop_i^t : \texttt{check } stan_s^t, lit_{ia}^t, poz_j^t; \texttt{set } lit_{ia}^{t+1}, poz_j^{t+1}; \texttt{jmp } split_{s_1 s_2}^{t+1};$

$split_{s_1 s_2}^{t+1} : \texttt{jmp } go_{s_1}^{t+1} \texttt{ and } go_{s_2}^{t+1};$

$go_{s_1}^{t+1} : \texttt{check } \varnothing; \texttt{set } stan_{s_1}^{t+1}; \texttt{jmp } loop_1^{t+1};$

Złożoność intuicjonistycznej logiki zdaniowej

Twierdzenie:

Intuicjonistyczny rachunek zdań jest PSPACE-zupełny.

Konstrukcja dowodu jako obliczenie

Konfigurację automatu reprezentujemy jako osąd $\Gamma \vdash p$ (gdzie p – atom).

- Cel atomowy p to stan maszyny;
- Otoczenie Γ to zawartość pamięci, w tym program.
- Założenie $p \to q$ umożliwia zmianę stanu z q na p.
- Założenie $(b \to p) \to q$ umożliwia zmianę stanu z q na p, z jednoczesnym zapisaniem b w pamięci.
- Założenie $a \to p \to q$ umożliwia zmianę stanu z q na p, pod warunkiem, że w pamięci jest zapisane a.

Uwaga: (1) nie można usunąć danej z pamięci;
(2) nie można sprawdzić, że danej w pamięci nie ma.

Programs into proofs!

Given $\mathcal{M} = \langle Q, R, f, \mathcal{I}\rangle$ and $C_0 = \langle q_0, S_0\rangle$,

define a set of axioms Γ so that

$\Gamma \vdash q_0$ iff C_0 is accepting.

Propositional variables: states and registers of $\mathcal{M}$.

Put all of S_0 into Γ, also $f \in \Gamma$.

Other axioms represent instructions of $\mathcal{M}$.

Instructions seen as axioms

Axiom for $q : \texttt{check } S_1; \texttt{set } S_2; \texttt{jmp } p$,
where $S_1 = \{s_1^1, ..., s_1^k\}$ and $S_2 = \{s_2^1, ..., s_2^\ell\}$:

(1) $s_1^1 \to \cdots \to s_1^k \to (s_2^1 \to \cdots \to s_2^\ell \to p) \to q.$

Axiom for $q : \texttt{jmp } p_1 \texttt{ and } p_2$:

(2) $p_1 \to p_2 \to q.$

Proof construction as computation

To prove $\Gamma, S \vdash q$, one can:

- Note that $q = f$ and observe $f \in \Gamma$;
- Use axiom $p_1 \to p_2 \to q$ and prove in parallel:
  $\Gamma, S \vdash p_1$ and $\Gamma, S \vdash p_2$
- Use axiom $s_1^1 \to \cdots \to s_1^k \to (s_2^1 \to \cdots s_2^\ell \to p) \to q$, provided $S_1 = \{s_1^1, ..., s_1^k\} \subseteq S$, and prove that
  $\Gamma, S, s_2^1, ..., s_2^\ell \vdash p$

The technical lemma we need

Lemma

$\Gamma, S \vdash q$ *iff* $\langle q, S_0 \cup S \rangle$ *is accepting.*

Proof.

($\Rightarrow$) Induction wrt the size of proof.

($\Leftarrow$) Induction wrt the definition of acceptance. $\square$