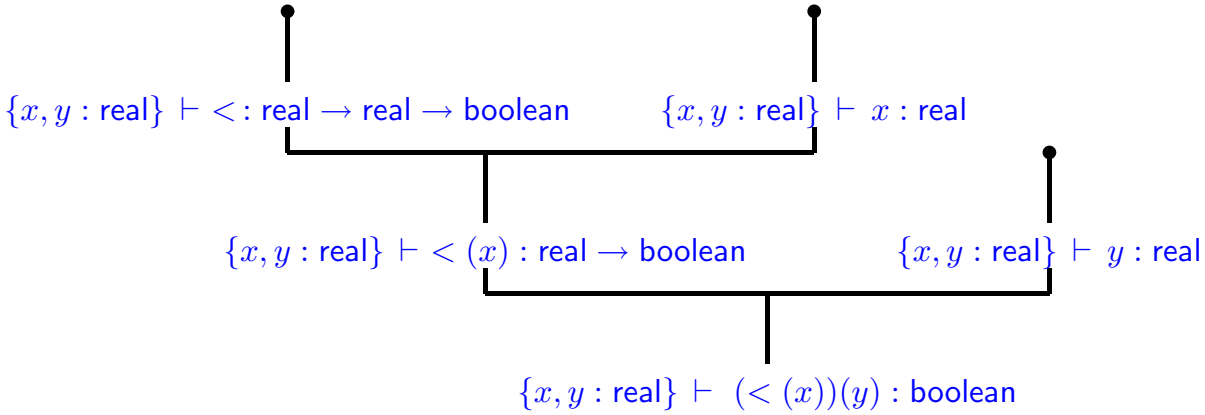
Reguły wnioskowania dla systemu typów



Zastosowanie funkcji $<$ do argumentu x daje w wyniku funkcję $< (x) : \text{real} \to \text{boolean}$ przypisującą dowolnemu argumentowi wynik porównania tego argumentu z x.

$(< (x))(y) : \text{boolean}$ jest wynikiem tego porównania — zwykle zapisywanym jako $x < y : \text{boolean}$.

Rachunek zdań a system typów

Rachunek zdań	Rachunek typów
Wprowadzanie koniunkcji: $\dfrac{\{\ldots\} \vdash q \quad \{\ldots\} \vdash r}{\{\ldots\} \vdash q \,\&\, r}$	Typowanie iloczynu kartezjańskiego: $\dfrac{\{\ldots\} \vdash t_1 : T_1 \quad \{\ldots\} \vdash t_2 : T_2}{\{\ldots\} \vdash \langle t_1, t_2 \rangle : T_1 \times T_2}$
Eliminacja koniunkcji: $\dfrac{\{\ldots\} \vdash q \,\&\, r}{\{\ldots\} \vdash q}$	Typowanie rzutowania: $\dfrac{\{\ldots\} \vdash t : T_1 \times T_2}{\{\ldots\} \vdash \text{fst}\, t : T_1}$
Wprowadzanie implikacji: $\dfrac{\{\ldots, q\} \vdash r}{\{\ldots\} \vdash q \Rightarrow r}$	Typowanie funkcji: $\dfrac{\{\ldots, x : T_1\} \vdash t : T_2}{\{\ldots\} \vdash (\lambda x.t) : T_1 \to T_2}$
Eliminacja implikacji: $\dfrac{\{\ldots\} \vdash p \Rightarrow q \quad \{\ldots\} \vdash p}{\{\ldots\} \vdash q}$	Typowanie zastosowania funkcji: $\dfrac{\{\ldots\} \vdash f : T_1 \to T_2 \quad \{\ldots\} \vdash t : T_1}{\{\ldots\} \vdash f(t) : T_2}$

Rachunek zdań a system typów

Wnioskowanie o koniunkcji $q \;\&\; r$ jest podobne do wnioskowania o iloczynie kartezjańskim $T_1 \times T_2$.

Wnioskowanie o implikacji $q \Rightarrow r$ jest podobne do wnioskowania o typie funkcji $T_1 \to T_2$.

Izomorfizm Curry'ego-Howarda:

zdania	odpowiadają	typom
dowody zdań	odpowiadają	elementom typów
zdania bez dowodów	odpowiadają	typom „niezamieszkałym"

Rachunek predykatów

$\forall_x\ p(x)$ — dla każdego x zachodzi $p(x)$

$\exists_x\ p(x)$ — istnieje takie x że zachodzi $p(x)$

$\forall_x\ x^2 \geqslant 0$

$\exists_x\ \forall_y\ x + y = y \qquad \neg\forall_x\ \neg\forall_y\ x + y = y$

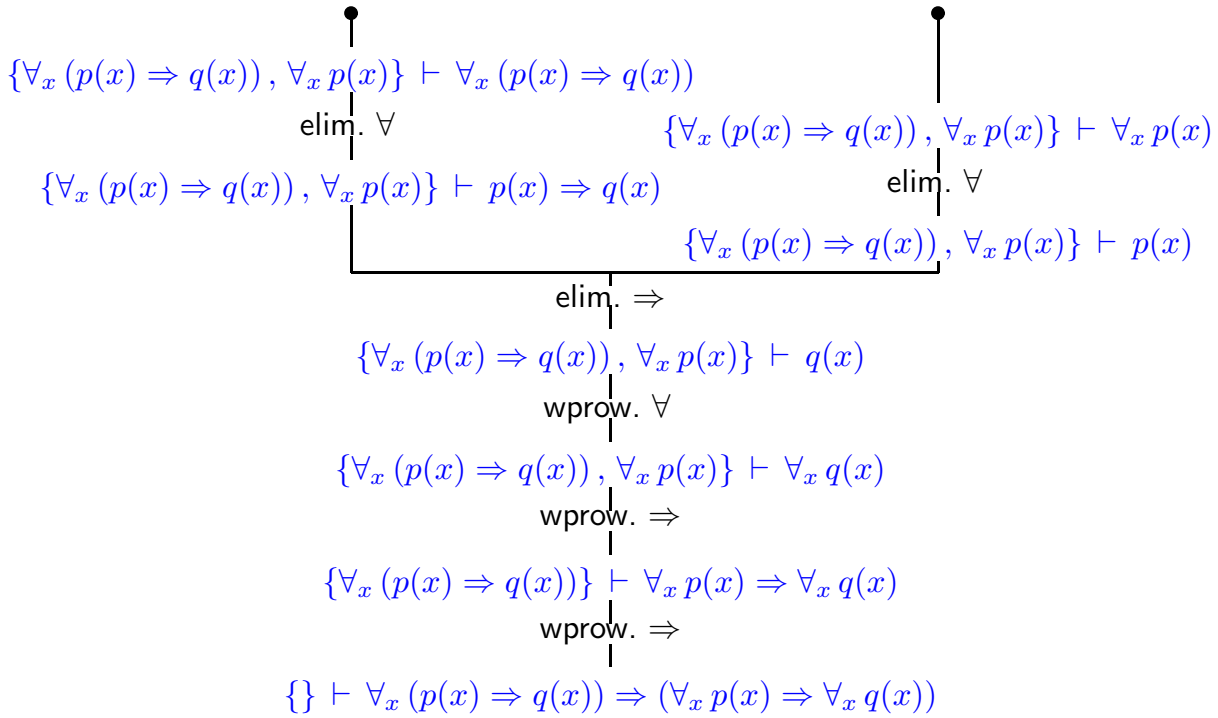
Wprowadzanie $\forall$:

$$\frac{\{\Gamma\} \vdash p(x)}{\{\Gamma\} \vdash \forall_x\ p(x)}$$ (o ile w Γ nie występuje wolny x)

Eliminacja $\forall$:

$$\frac{\{\ldots\} \vdash \forall_x\ p(x)}{\{\ldots\} \vdash p(t)}$$

Rachunek predykatów



Rachunek predykatów

Przykład: Złe rozumowanie:

	z założenia:	$\{x \geqslant 0\} \vdash x \geqslant 0$
ŹLE! $\longrightarrow$	wprowadzenie $\forall$:	$\{x \geqslant 0\} \vdash \forall_x\, x \geqslant 0$
	wprowadzenie $\Rightarrow$:	$\{\} \vdash x \geqslant 0 \Rightarrow \forall_x\, x \geqslant 0$
	wprowadzenie $\forall$:	$\{\} \vdash \forall_x (x \geqslant 0 \Rightarrow \forall_x\, x \geqslant 0)$
	eliminacja $\forall$ (0 za x):	$\{\} \vdash 0 \geqslant 0 \Rightarrow \forall_x\, x \geqslant 0$
	eliminacja $\Rightarrow$ (bo $0 \geqslant 0$ jest prawdą):	$\{\} \vdash \forall_x\, x \geqslant 0$
	eliminacja $\forall$ (-1 za x):	$\{\} \vdash -1 \geqslant 0$

Rodzaje języków programowania:

- **imperatywne** — program składa się z instrukcji „najpierw zrób to, potem tamto" (Pascal, C, ...); oparte na *maszynie Turinga*;
- **funkcyjne** lub **funkcjonalne** — program składa się z definicji i wyrażeń zawierających zdefiniowane terminy (LISP, SML, Haskell, ...); oparte na *λ-rachunku*;
- **logiczne** — program składa się z aksjomatów, do których system szuka kontrprzykładu (PROLOG); oparte na *metodzie rezolucji w logice*;
- **obiektowe** — program składa się z obiektów wywołujących swoje metody (Smalltalk, Java, częściowo C++, ...); oparte na *teorii algebr*.