Logika pre informatikov 2

Paul J. Voda   Ján Kľuka

Katedra aplikovanej informatiky
FMFI UK v Bratislave
http://ii.fmph.uniba.sk/cl/courses/lpi2

Propositional Logic

Lecture 1

The language of propositional logic

Propositional formulas are formed from

- **propositional variables** $(P_0, P_1, \ldots)$ by
- **propositional connectives** which are
  - **nullary**: truth $(\top)$, falsehood $(\bot)$
  - **unary**: negation $(\neg)$
  - **binary**: disjunction $(\vee)$, conjunction $(\wedge)$
    implication $(\rightarrow)$, equivalence $(\leftrightarrow)$

Binary are **infix** ($\rightarrow, \leftrightarrow$ groups to the right, the rest to the left)

Precedence from highest is $\neg$, $\wedge$, $\vee$, $(\rightarrow, \leftrightarrow)$. Thus
$P_1 \rightarrow P_2 \leftrightarrow P_3 \vee \neg P_4 \wedge P_5$ abbreviates
$P_1 \rightarrow (P_2 \leftrightarrow (P_3 \vee (\neg(P_4) \wedge P_5)))$

Truth functions

We identify the **truth values** *true* and *false* with the nullary symbols $\top$ and $\bot$ respectively.
The remaining connectives are **interpreted** as functions over truth values satisfying:

P_1	P_2	$\neg P_1$	$P_1 \wedge P_2$	$P_1 \vee P_2$	$P_1 \rightarrow P_2$	$P_1 \leftrightarrow P_2$
$\bot$	$\bot$	$\top$	$\bot$	$\bot$	$\top$	$\top$
$\bot$	$\top$	$\top$	$\bot$	$\top$	$\top$	$\bot$
$\top$	$\bot$	$\bot$	$\bot$	$\top$	$\bot$	$\bot$
$\top$	$\top$	$\bot$	$\top$	$\top$	$\top$	$\top$

We have

$$A \leftrightarrow B \equiv (A \rightarrow B) \wedge (B \rightarrow A)$$
$$\neg A \equiv A \rightarrow \bot$$
$$A \rightarrow B \equiv \neg A \vee B$$
$$A \wedge B \equiv \neg(\neg A \vee \neg B)$$

Tautologies

Of special interest are those propositional formulas A which are true ($\top$) for all possible truth values of its propositional variables, in writing $\models_p A$.

Every such formula is a **tautology**.

Tautologies are the cornerstones of mathematical logic.

Some examples of (schemas of) tautologies:

$$\models_p (A \to B \to C) \leftrightarrow A \land B \to C$$
$$\models_p (A \to B \to C) \leftrightarrow (A \to B) \to A \to C$$
$$\models_p (A \to B) \leftrightarrow \neg B \to \neg A$$

for any propositional formulas A, B, and C

Propositional satisfaction relation

A **propositional valuation**, or an **propositional assignment** v is a (possibly infinite) set $v \subset \mathbb{N}$ The idea is that the $P_i \equiv \top$ iff $i \in v$.

We say that a formula A **is satisfied in** v, in writing $\models_p^v A$, if A is true for the assignment v.
We thus have:
$\models_p^v P_i$ iff $i \in v$
$\models_p^v \neg A$ iff not $\models_p^v A$ iff $\not\models_p^v A$
$\models_p^v A \wedge B$ iff $\models_p^v A$ and $\models_p^v B$
$\models_p^v A \vee B$ iff $\models_p^v A$ or $\models_p^v B$
$\models_p^v A \rightarrow B$ iff whenever $\models_p^v A$ also $\models_p^v B$

Thus A **is a tautology** iff $\models^v A$ for all valuations v.

Coincidence property if two valuations v and w are such that $i \in v$ iff $i \in w$ for all P_i occurring in A then $\models_p^v A$ iff $\models_p^w A$

Satisfaction relation for sets of propositional formulas

For T a set of formulas and v a valuation (both possibly infinite), we say that v **satisfies** T, in writing $\models_p^v T$, iff for all $A \in T$ we have $\models_p^v A$.

We say that S **is a propositional (tautological) consequence of** T, in writing $T \models_p S$, iff for all v satisfying T (i.e. $\models_p^v T$) at least one $A \in S$ is satisfied (i.e. $\models_p^v A$)

The special case when $T \models_p \{A\}$ is the most important relation in mathematical logic. We write $T \models_p A$ instead of $T \models_p \{A\}$ and say that A **tautologically follows from** T. Note that $\emptyset \models_p \{A\}$ iff A is tautology.

If $T \models_p S$ holds then we say that the **propositional sequent** $T \models_p S$ is valid

Compactness theorem for propositional consequence

$T \models_p S$ iff there are finite $T' \subset T$ and $S' \subset S$ s.t. $T' \models_p S'$.

If $T' = \{A_1, \ldots, A_n\}$ and $S' = \{B_1, \ldots, B_m\}$ we have $T' \models_p S'$ iff

$$\models_p A_1 \wedge \cdots \wedge A_n \to B_1 \vee \cdots \vee B_m$$

Saturation of propositional sequents

Closure:

$A, T \models_p A, S$; $\bot, T \models_p S$; $T \models_p \top, S$ are valid

Flattenings:

- $T \models_p A \rightarrow B, S$ iff $A, T \models_p B, A \rightarrow B, S$
- $T \models_p A \vee B, S$ iff $T \models_p A, B, A \vee B, S$
- $A \wedge B, T \models_p S$ iff $A, B, A \wedge B, T \models_p S$

Splits:

- $A \rightarrow B, T \models_p S$ iff
$B, A \rightarrow B, T \models_p S$ and $A \rightarrow B, T \models_p A, S$
- $A \vee B, T \models_p S$ iff
$A, A \vee B, T \models_p S$ and $B, A \vee B, T \models_p S$
- $T \models_p A \wedge B, S$ iff
$T \models_p A, A \wedge B, S$ and $T \models_p B, A \wedge B, S$

Inversions:

- $T \models_p \neg A, S$ iff $A, T \models_p \neg A, S$
- $\neg A, T \models_p S$ iff $\neg A, T \models_p A, S$

Cuts:

$T \models_p S$ iff $A, T \models_p S$ and $T \models_p A, S$

Here $A_1, \ldots, A_k$ stands for $A_1, \ldots, A_k, \emptyset$ and

$A_1, \ldots, A_k, S$ stands for $S \cup \{A_1, \ldots, A_k\}$.

Propositional tableaux as trees of sequents

A branch with formulas $A_1, \ldots, A_n, B_1*, \ldots, B_m*$ can be viewed as a finite *sequent*:
$\{A_1, \ldots, A_n\} \vDash_p \{B_1, \ldots B_m\}$ A tableau can be viewed as a conjunction of sequents corresponding to its branches.
Tableau rules: correspond to saturation of sequents:
A branch **closes** when it contains $\bot$, $\top*$, $A, A*$
Flattens:

$$\frac{A \to B*}{A,\ B*} \quad \frac{A \vee B*}{A*,\ B*} \quad \frac{A \wedge B}{A,\ B}$$

Splits:

$$\frac{A \to B}{B \mid A*} \quad \frac{A \vee B}{A \mid B} \quad \frac{A \wedge B*}{A* \mid B*}$$

Inversions, Cuts, and Axioms:

$$\frac{\neg A*}{A} \quad \frac{\neg A}{A*} \quad \frac{}{A \mid A*} \quad \frac{}{A} \text{ when } A \in T$$

Saturated sequents

A sequent $T \vDash_p S$ (a branch of a tableau) is **saturated** if no rule can be applied to it, i.e.

- if $A \rightarrow B \in S$ then $A \in T$ and $B \in S$
- if $A \vee B \in S$ then $A \in S$ and $B \in S$
- it $A \wedge B \in T$ then $A \in T$ and $B \in T$

- if $A \rightarrow B \in T$ then $B \in T$ or $A \in S$
- if $A \vee B \in T$ then $A \in T$ or $B \in T$
- if $A \wedge B \in S$ then $A \in S$ or $B \in S$

- if $\neg A \in S$ then $A \in T$
- if $\neg A \in T$ then $A \in S$

If a saturated sequent is closed then it is valid because it cannot be falsified in any v.

If a saturated sequent is valid then it is closed, because if not closed then $v = \{i \mid P_i \in T\}$ falsifies the sequent.

Soundness and completeness of propositional tableaux

We write $T \vDash_p S \rhd_p T' \vDash_p S'$ when the first sequent is a **father** of the second one.
We have $T \vDash_p S$ iff $T' \vDash_p S'$ for all saturated sons.

When we write $T \vdash_p A$ for **there is a closed tableau for the goal** A then we have

Soundness: if $T \vdash_p A$ then $T \vDash_p A$

Completeness: if $T \vDash_p A$ then $T \vdash_p A$

Equational Logic

Lecture 2

Language of equational logic

$\mathcal{L}$ consists of **terms** given by **denumerable** sets of **function symbols** f_i and **predicate symbols** P_i (each with *arity* $n \geq 0$). We always have $=$ among predicate symbols.

Terms: are concrete sequences of symbols defined by:

1. (object) variables $x_0, x_1, \ldots, y_0, y_1, \ldots$ are terms,
2. if $\tau_1, \ldots, \tau_n$ are terms and the function symbol f_i has arity $n \geq 0$ then $f_i(\tau_1, \ldots, \tau_n)$ is a term.

Function symbols of arity 0 are *constants*

Formulas are concrete sequences of symbols consisting of:

1. *atomic* formulas $P_i(\tau_1, \ldots, \tau_n)$ with $\tau_1, \ldots, \tau_n$ terms,
2. *propositional* formulas $\bot$, $\top$, $\neg A_1$, $A_1 \vee A_2$, $A_1 \wedge A_2$, $A_1 \rightarrow A_2$, $A_1 \leftrightarrow A_2$ with A_1, A_2 formulas,

We write $\tau_1 = \tau_2$ for **identities** $=(\tau_1, \tau_2)$.

Predicate symbols of arity 0 are *propositional constants* and they correspond to propositional variables.

Quasi-tautological consequence

We wish to define **equationally valid** sequents $T \models_i S$ such that if $T \models_p S$ then $T \models_i S$ (i.e. all tautologies are eq. valid). But we also wish to use the properties of $=$, for instance, $\models_i \tau_1 = \tau_2 \rightarrow \tau_2 = \tau_1$ which is not in general a tautology but it is a **quasi-tautology**.

If $T \models_i A$ we say that A is a **quasi-tautological consequence** of T.

We wish the **reduction** to propositional logic:

$$T \models_i S \text{ iff } T, \mathbf{Eq} \models_p S$$

where **Eq** are the **axioms of identity**.

Identity (equational) axioms

for every equational language $\mathcal{L}$ the set of sentences $\tau = \tau$ are **reflexivity** axioms,

$$\tau = \sigma \rightarrow \sigma = \tau$$

are **symmetry** axioms,

$$\tau = \sigma \rightarrow \sigma = \rho \rightarrow \tau = \rho$$

are **transitivity** axioms, and

$$\tau_1 = \sigma_1 \rightarrow \cdots \rightarrow \tau_n = \sigma_n \rightarrow f(\tau_1, \ldots, \tau_n) = f(\sigma_1, \ldots, \sigma_n)$$

$$\tau_1 = \sigma_1 \rightarrow \cdots \rightarrow \tau_n = \sigma_n \rightarrow P(\tau_1, \ldots, \tau_n) \rightarrow P(\sigma_1, \ldots, \sigma_n)$$

are **substitution** axioms where $f, P \in \mathcal{L}$

We designate all by **Eq** and call them **equation** (identity) axioms (for $\mathcal{L}$).

Interpretation of languages of identity

We wish to introduce **interpretations** $\mathcal{M}$ of $\mathcal{L}$ corresponding to propostional valuations v such that $\models_i^{\mathcal{M}} A$ if A **is satisfied in** $\mathcal{M}$.

This should extend propositional valuations, for instance, we wish:

- $\models_i^{\mathcal{M}} A \land B$ iff $\models_i^{\mathcal{M}} A$ and $\models_i^{\mathcal{M}} B$
- $\models_i^{\mathcal{M}} \neg A$ iff not $\models_i^{\mathcal{M}} A$

But we also wish $\models_i^{\mathcal{M}}$ **Eq**, i.e. for instance:

- if $\models_i^{\mathcal{M}} \tau_1 = \tau_2$ then $\models_i^{\mathcal{M}} \tau_2 = \tau_1$

Interpretations $\mathcal{M}$ for L

consist of **domains** D, of interpretions $f^{\mathcal{M}}$ of functions symbols $f \in \mathcal{L}$ as n-ary functions over D and of interpretions $P^{\mathcal{M}}$ of predicate symbols $P \in \mathcal{L}$ as n-ary relations over D.
$\langle D, \dots f^{\mathcal{M}}, \dots P^{\mathcal{M}}, \dots \rangle$ are **structures for** $\mathcal{L}$.
In **interpretations** $\mathcal{M}$ we also need to **assign objects** $x^{\mathcal{M}}$ from D to the (object) variables x.
Terms τ of $\mathcal{L}$ are *interpreted* in $\mathcal{M}$ by **denotations** $\tau^{\mathcal{M}} \in D$ s.t.

- $f(\tau_1, \dots, \tau_n)^{\mathcal{M}} = f^{\mathcal{M}}(\tau_1^{\mathcal{M}}, \dots, \tau_n^{\mathcal{M}})$.

Atomic formulas are interpreted by defining:

- $\models_i^{\mathcal{M}} P(\tau_1, \dots, \tau_n)$ iff $P^{\mathcal{M}}(\tau_1^{\mathcal{M}}, \dots, \tau_n^{\mathcal{M}})$,
- $\models_i^{\mathcal{M}} \tau_1 = \tau_2$ iff $\tau_1^{\mathcal{M}} = \tau_2^{\mathcal{M}}$

and we close the satisfaction relation propositionally.

Saturation of identity sequents

We **define** $T \models_i S$ to hold iff for all interpretations $\mathcal{M}$ satisfying T, i.e. $\models_i^{\mathcal{M}} T$, there is an $A \in S$ s.t. $\models_i^{\mathcal{M}} A$. Thus $\models_i A$, i.e. A is a **quasi-tautology**, iff $\models_i^{\mathcal{M}} A$ for all $\mathcal{M}$. We have

- $T \models_i S$ iff $\tau = \tau, T \models_i S$
- $\tau_1 = \tau_2, T \models_i S$ iff $\tau_2 = \tau_1, \tau_1 = \tau_2, T \models_i S$
- $\tau_1 = \tau_2, \tau_2 = \tau_3, T \models_i S$ iff $\tau_1 = \tau_3, \tau_1 = \tau_2, \tau_2 = \tau_3, T \models_i S$
- $\vec{\tau} = \vec{\rho}, T \models_i S$ iff $f(\vec{\tau}) = f(\vec{\rho}), \vec{\tau} = \vec{\rho}, T \models_i S$
- $\vec{\tau} = \vec{\rho}, P(\vec{\tau}), T \models_i S$ iff $P(\vec{\rho}), \vec{\tau} = \vec{\rho}, P(\vec{\tau}), T \models_i S$

plus all saturations corresponding to the propositional ones.

Tableau rules for identity

reflexivity rules:

$$\frac{}{\tau = \tau}$$

symmetry rules:

$$\frac{\tau = \sigma}{\sigma = \tau}$$

transitivity rules:

$$\frac{\tau = \sigma \quad \sigma = \rho}{\tau = \rho}$$

substitution rules:

$$\frac{\tau_1 = \sigma_1 \cdots \tau_n = \sigma_n}{f(\tau_1, \ldots, \tau_n) = f(\sigma_1, \ldots, \sigma_n)} \quad f \in \mathcal{L}$$

$$\frac{\tau_1 = \sigma_1 \cdots \tau_n = \sigma_n \; P(\tau_1, \ldots, \tau_n)}{P(\sigma_1, \ldots, \sigma_n)} \quad P \in \mathcal{L}$$

Equationally saturated sequents

$T \models_i S$ is **equationally saturated** if it is propositionally saturated and

- $\tau = \tau \in T$
- if $\tau_1 = \tau_2 \in T$ then $\tau_2 = \tau_1 \in T$
- if $\tau_1 = \tau_2, \tau_2 = \tau_3 \in T$ then $\tau_1 = \tau_3 \in T$
- if $\vec{\tau} = \vec{\rho} \in T$ then $f(\vec{\tau}) = f(\vec{\rho}) \in T$
- if $\vec{\tau} = \vec{\rho}, P(\vec{\tau}) \in T$ then $P(\vec{\rho}) \in T$

We have $T \models_i S$ iff all equationally saturated sons are closed.

Ekvačná a kvantifikačná logika

3. prednáška (6. 10. 2004)

Vhodnosť a úplnosť ekvačných tabiel

- Formula A je **ekvačne dokázateľná** z množiny axióm T ($T \vdash_i A$) práve vtedy, keď existuje uzavreté tablo pre cieľ A
- **Vhodnosť a úplnosť ekvačných tabiel** (soundness & completeness)

$$T \vdash_i A \text{ práve vtedy, keď } T \models_i A$$

$\Leftarrow$ Sporom:
Predpokladáme $T \nvdash_i A$ a skonštruujeme interpretáciu $\mathcal{M}$ **zo syntaktického materiálu (herbrandovskú)** tak, aby $\models_i^{\mathcal{M}} T$, ale $\not\models_i^{\mathcal{M}} A$

Redukcia ekvačnej logiky do propozičnej

- Syntaktická redukcia

$$T \vdash_i A \text{ práve vtedy, keď } T, \mathbf{Eq} \vdash_p A$$

- Sémantická redukcia

$$T \models_i A \text{ práve vtedy, keď } T, \mathbf{Eq}_{A,T} \models_p A$$

$$T \models_i A \iff T \vdash_i A \iff T, \mathbf{Eq} \vdash_p A \iff T, \mathbf{Eq} \models_p A$$

Jazyk kvantifikačnej logiky

- **Termy** ako v ekvačnej logike
- **Formuly**
  - **atomické**: $P_i(\tau_1, \ldots, \tau_n)$, ak $\tau_1, \ldots, \tau_n$ sú termy a P_i je predikátový symbol arity n
  - **propozičné**: $\bot$, $\top$, $\neg A_1$, $A_1 \land A_2$, $A_1 \lor A_2$, $A_1 \rightarrow A_2$, $A_1 \leftrightarrow A_2$, ak A_1 a A_2 sú formuly
  - **kvantifikačné**:
    - **existenčné**: $\exists x\, A[x]$,
    - **všeobecné**: $\forall x\, A[x]$,

    ak $A[x]$ je formula, v ktorej sa môže vyskytovať objektová premenná x

Voľné a viazané premenné

- Premenná x je **viazaná** (**bound**) vo formulách $\exists x\, A[x]$ a $\forall x\, A[x]$
- Premenné, ktoré nie sú viazané, sú **voľné** (**free**)

$$\exists z\, x = f(z) \rightarrow \forall y(P(y, x) \vee \exists x\, y = g(x, x))$$

- Dosadenie: Ak $A[x]$ je formula, v ktorej sa môže vyskytovať premenná x, $A[\tau]$ vznikne dosadením termu τ za voľné výskyty x
- $\tau \equiv ff(7)$

$$\exists z\, ff(7) = f(z) \rightarrow \forall y(P(y, ff(7)) \vee \exists x\, y = g(x, x))$$

- Formula bez voľných premenných sa nazýva **veta** (**sentence**)

Interpretácie a spĺňanie formúl

- **Štruktúry** a **interpretácie** ako v ekvačnej logike
- Relácia $\models^{\mathcal{M}} A$:
  formula A je **splnená** (**satisfied**; tiež **pravdivá**) v interpretácii $\mathcal{M}$
  - pre atomické A ako $\models_i^{\mathcal{M}} A$
  - $\models^{\mathcal{M}} \exists x\, A[x]$ práve vtedy, keď $\models^{\mathcal{M}'} A[y]$ platí pre aspoň jedno **rozšírenie** (**expansion**) $\mathcal{M}'$ interpretácie $\mathcal{M}$ o intepretáciu novej premennej y
  - $\models^{\mathcal{M}} \forall x\, A[x]$ práve vtedy, keď $\models^{\mathcal{M}'} A[y]$ platí pre všetky rozšírenia $\mathcal{M}'$ interpretácie $\mathcal{M}$ o intepretáciu novej premennej y
  - pre propozičné A ($\neg A_1$, $A_1 \wedge A_2$, $\dots$) analogicky ako propozičná pravdivosť
- Interpretácia $\mathcal{M}$ spĺňa množinu formúl T ($\models^{\mathcal{M}} T$) práve vtedy, keď pre všetky formuly $A \in T$ platí $\models^{\mathcal{M}} A$

Logické vyplývanie a platné formuly

- Množina formúl S **logicky vyplýva** z množiny formúl T (S is a **logical consequence** of T; $T \models S$) práve vtedy, keď v každej interpretácii $\mathcal{M}$ spĺňajúcej T platí $\models^{\mathcal{M}} A$ pre aspoň jedno $A \in S$
- Formula A je **platná** (**valid**; $\models A$), keď $\emptyset \models \{A\}$, teda keď je A splnená v každej interpretácii

Tautológie, kvázitautológie a platné formuly

- Všetky tautológie sú kvázitautológiami ($\models_p A \implies \models_i A$)
- Všetky kvázitautológie sú platnými formulami ($\models_i A \implies \models A$)
- **Ale nie naopak!**
- $x = y \rightarrow y = x$: kvázitautológia, ale nie tautológia
  Z pohľadu propozičnej logiky sú atomické a kvantifikačné formuly **propozičnými premennými**
- $\forall x(a = b \land b = c) \rightarrow a = c$:
  platná formula, ale nie kvázitautológia
  Z pohľadu ekvačnej logiky sú kvantifikačné formuly **propozičnými konštantami**

Saturácia kvantifikačných sekventov

Logické vyplývanie $T \models S$ má nasledujúce vlastnosti, ktoré nám umožnia **saturovať** množiny T a S.

- $\forall x\, A[x], T \models S$ práve vtedy, keď $A[\tau], \forall x\, A[x], T \models S$ pre ľubovoľný term τ
- $T \models \exists x\, A[x], S$ práve vtedy, keď $T \models A[\tau], \exists x\, A[x], S$ pre ľubovoľný term τ
- $\exists x\, A[x], T \models S$ práve vtedy, keď $A[z], \exists x\, A[x], T \models S$ pre **novú premennú** z (nie je voľná v T ani v S)
- $T \models \forall x\, A[x], S$ práve vtedy, keď $T \models A[z], \forall x\, A[x], S$ pre **novú premennú** z
- Všetky saturačné vlastnosti kvázitautologického a propozičného vyplývania

Tablové pravidlá pre kvantifikátory

- Inštanciačné pravidlá (instantiation rules)

$$\frac{\forall x\, A[x]}{A[\tau]}(\forall), \qquad \frac{\exists x\, A[x]*}{A[\tau]*}(\exists*)$$

pre ľubovoľný term τ

- Pravidlá s vlastnou premennou (eigen-variable rules)

$$\frac{\exists x\, A[x]}{A[z]}(\exists), \qquad \frac{\forall x\, A[x]*}{A[z]*}(\forall*),$$

kde z je premenná, ktorá sa v table ešte nevyskytla voľná

Kvantifikačné axiómy

Kvantifikačným pravidlám zodpovedajú kvantifikačné axiómy

- Inštanciačné axiómy

$$\forall x\, A[x] \rightarrow A[\tau]$$
$$A[\tau] \rightarrow \exists x\, A[x]$$

pre ľubovoľný term τ

- Axiómy s vlastnými premennými
  - dosvedčujúca (**witnessing**)

$$\exists x\, A[x] \rightarrow A[z]$$

vlastná premenná z sa nazýva svedok (**witness**)

  - kontrapríkladová

$$A[z] \rightarrow \forall x\, A[x]$$

vlastná premenná z sa nazýva kontrapríklad (**counterexample**)

pre „vhodnú" premennú z

Regulárne množiny kvantifikačných axióm

Požiadavky na vlastné premenné

- Vlastná premenná axiómy $\exists x\, A[x] \to A[z]$ sa nesmie vyskytovať v $\exists x\, A[x]$
- Vlastná premenná axiómy $A[z] \to \forall x\, A[x]$ sa nesmie vyskytovať v $\forall x\, A[x]$
- Všetky kvantifikačné axiómy použité v table musia tvoriť **regulárnu množinu**, teda je možné usporiadať ich do postupnosti

$$\langle Q_1, Q_2, \ldots, Q_n \rangle$$

tak, že **vlastná premenná** axiómy Q_{i+1}, $i < n$, **sa nevyskytuje** v žiadnej axióme Q_j pre $1 \le j \le i$

Význam kvantifikačných axióm

Kvantifikačné axiómy umožnia redukovať kvantifikačnú logiku do ekvačnej:

$$T \models S \text{ práve vtedy, keď } T, \mathbf{Q} \models_i S$$

pre vhodne zvolenú množinu kvantifikačných axióm $\mathbf{Q}$

Kvantifikačná logika (dokončenie)

4. prednáška (13. 10. 2004)

Vhodnosť kvantifikačných tabiel

- Formula A je **kvantifikačne dokázateľná** z množiny axióm T ($T \vdash A$) práve vtedy, keď existuje uzavreté kvantifikačné tablo pre cieľ A
- **Vhodnosť kvantifikačných tabiel** (soundness)

$$\text{Ak } T \vdash A, \text{ potom } T \models A$$

- Indukciou na konštrukciu tabla z vlastností logického vyplývania

Kvantifikačne saturované sekventy

Sekvent $T \models S$ (vetva tabla) je **kvantifikačne saturovaný**, ak je ekvačne saturovaný a navyše

- ak $\forall x\, A[x] \in T$, tak $A[\tau] \in T$ pre každý term τ,
- ak $\exists x\, A[x] \in S$, tak $A[\tau] \in S$ pre každý term τ,
- ak $\exists x\, A[x] \in T$, tak $A[z] \in T$ pre aspoň jednu premennú z,
- ak $\forall x\, A[x] \in S$, tak $A[z] \in S$ pre aspoň jednu premennú z.

Úplnosť kvantifikačných tabiel

- Gödelova veta o úplnosti (completeness)

$$\text{Ak } T \models A, \text{ tak } T \vdash A$$

- Nepriamo:
  - Predpokladáme $T \nvdash A$
  - Každé tablo má otvorenú vetvu
  - Skonštruujeme systematické tablo
  - Otvorené vetvy sú saturované
  - Špeciálny prípad:
    konečná vetva saturovaná vzhľadom na vlastné premenné
  - Z otvorenej vetvy skonštruujeme interpretáciu $\mathcal{M}$ (zo syntaktického materiálu) spĺňajúcu všetky predpoklady (teda aj T), ale žiaden cieľ (teda ani A), t. j. $\models^{\mathcal{M}} T$ a $\not\models^{\mathcal{M}} A$, preto $T \not\models A$

Redukcia kvantifikačnej logiky do ekvačnej a propozičnej

- Syntaktická redukcia:
  $T \vdash A$ práve vtedy, keď existuje regulárna množina $\mathbf{Q}$, pre ktorú $T, \mathbf{Q} \vdash_i A$
- Sémantická redukcia:
  Existuje regulárna množina $\mathbf{Q}$ taká, že

  $$T \models A \text{ práve vtedy, keď } T, \mathbf{Q} \models_i A$$

- Redukcia do propozičnej logiky:

  $$T \models A \text{ práve vtedy, keď } T, \mathbf{Eq}, \mathbf{Q} \models_p A$$

Extension of theories

Lecture 5

Recapitulation of Predicate logic

- **Gödel's completeness and soundness**:

$$T \models A \text{ iff } T \vdash A$$

- **Reduction of predicate logic to propositional:**

$$T \models A \text{ iff } T, \mathbf{Eq}, \mathbf{Q} \models_p A$$

Extensions of theories

- **Extension of languages**: $\mathcal{L}'$ is an **extension** of $\mathcal{L}$ if every symbol of $\mathcal{L}$ is a symbol of $\mathcal{L}'$,
- **Extension of theories**: T' is an **extension** of T if the language of T' extends the language of T and $T' \vdash A$ for all $A \in T$,
- **Conservative extensions:** An extension T' of T is **conservative** iff from $T' \vdash A$ where A is in the language of T we have $T \vdash A$
- **Consistent theories:** A theory T is **consistent** if $T \nvdash \bot$. Clearly, if T' is conservative over a consistent T then also T' is consistent

Extension by definitions with predicate symbols

- Let T be a theory in $\mathcal{L}$ which does not contain n-ary predicate symbol P, and $A[\vec{x}]$ a formula of $\mathcal{L}$ with just the n-variables $\vec{x}$ free,
- then $T' = T + \forall\vec{x}(P(\vec{x}) \leftrightarrow A[\vec{x}])$ is an **extension** of T in the language $\mathcal{L} + P$,
- **Elimination of** P: Let B^* be like B but with every $P(\vec{\tau})$ replaced by $A[\vec{\tau}]$,
- $T' \vdash B \leftrightarrow B^*$, proof is straightforward,
- T' is **conservative over** T: If $T' \vDash B \in \mathcal{L}$ take any $\vDash^{\mathcal{M}} T$, expand it to $\vDash^{\mathcal{M}'} T'$, conclude $\vDash^{\mathcal{M}'} B$, and $\vDash^{\mathcal{M}} B$. Hence $T \vDash B$
- **Translation:** $T' \vdash B$ iff $T \vdash B^*$ for any $B \in \mathcal{L} + P$

Extension by definitions with function symbols

- Let T be a theory in $\mathcal{L}$ which does not contain n-ary function symbol f, and $A[\vec{x}, y]$ a formula of $\mathcal{L}$ with just the $n+1$-variables $\vec{x}, y$ free,
- if the **existence condition**: $T \vdash \exists y A[\vec{x}, y]$ holds then
- $T' = T + A[\vec{x}, f(\vec{x})]$ is **conservative** over T: If $T' \vDash B \in \mathcal{L}$ take any $\vDash^{\mathcal{M}} T$, expand it to $\vDash^{\mathcal{M}'} T'$, conclude $\vDash^{\mathcal{M}'} B$, and $\vDash^{\mathcal{M}} B$. Hence $T \vDash B$
- if also the **uniqueness condition**: $T \vdash A[\vec{x}, y_1] = A[\vec{x}, y_2] \rightarrow y_1 = y_2$ holds
- then $T'' = T + \forall \vec{x}(f(\vec{x}) = y \leftrightarrow A[\vec{x}, y])$ is **conservative over** T because T' extends T''.
- **Elimination of** f: Let B^* be like B but with every atomic subformula $C_y[f(\vec{\tau})]$ replaced by $\exists y(A[\vec{\tau}, y] \wedge C[y])$,
- $T'' \vdash B \leftrightarrow B^*$, proof is straightforward,
- **Translation:** $T'' \vdash B$ iff $T \vdash B^*$ for any $B \in \mathcal{L} + P$

Basic bootstrapping of PA

Lecture 6

Recapitulation of extensions

- T' is an **extension** of T if $T' \models A$ for all $A \in T$ (T' **can** prove new facts about formulas of $\mathcal{L}_T$)
- T' is a **conservative extension of** T if for all $A \in \mathcal{L}_T$ from $T' \models A$ we get $T \models A$ (T' **cannot** prove new facts about formulas of $\mathcal{L}_T$ but it can about formulas of $\mathcal{L}_{T'}$),
- Special case of conservative extensions are **extensions by definitions** where **no** new facts about formulas of $\mathcal{L}_{T'}$ can be proved because every theorem of T' can be **translated** to an equivalent theorem of T.

Goals

- For arbitrary **theory** T we have learnt to prove **theorems** A of **extensions** T' as **logical consequences**: $T' \models A$,
- we will now study a particular theory **Peano arithmetic** (PA)
- our **goal** is to show that the **clauses** of legal **CL** definitions are theorems in **definitional extensions** of PA
- Thus all properties of CL programs are provable in PA but the extensions make for **readability** and for the **computability** directly from the clauses

Peano arithmetic

The **language of PA** consists of the **constant** 0 and **function symbols** x', $x + y$, $x \cdot y$.
The **standard structure** $\mathcal{N}$ has the domain $\mathbb{N}$ of natural numbers with the intended interpretaion of symbols in that order as **zero**, **successor**, **addition**, and **multiplication** functions.
The **axioms of PA** are

$$x' \neq 0 \qquad x' = y' \rightarrow x = y$$
$$0 + y = y \qquad x' + y = (x + y)'$$
$$0 \cdot y = 0 \qquad x' \cdot y = (x \cdot y) + y$$
$$A[0, \vec{y}] \land \forall x (A[x, \vec{y}] \rightarrow A[x', \vec{y}]) \rightarrow A[x, \vec{y}]$$

for **all** formulas $A[x, \vec{y}]$ of the language of PA. The last axioms are called the axioms of **mathematical induction** with x called the **induction** variable and $\vec{y}$ (if any) the **parameters**

Incompleteness of PA: Goodstein's sequence

For $x > 0$ write the number $x - 1$ **fully** in **base** $n \geq 2$. For instance, for $x = 528$ and $n = 2$:
$x = 528 = 2^9 + 2^4 + 1 = 2^{2^3+1} + 2^{2^2} = 2^{2^{2^1+1}} + 2^{2^{2^1}}$
$x = 527 = 2^{2^{2^1+1}} + 2^{2^1+1} + 2^{2^1} + 2^1 + 1$ and **change** to base $n + 1 = 3$:
$P_n(x) = 3^{3^{3^1+1}} + 3^{3^1+1} + 3^{3^1} + 3^1 + 1.$
Subtract one and change to base 4, obtain $P_{n+1}(P_n(x))$, and continue. This is called **Goodstein's sequence**
There is a formula $A[n, x]$ of PA which says **Goodstein's sequence for $n \geq 2$ and any x terminates in finitely many steps in** 0
We have $\models^{\mathcal{N}} \forall n \forall x A[n, x]$ **but** PA $\not\vdash \forall n \forall x A[n, x]$.
Hence by **Gödel's completeness** there is a **non-standard** structure $\mathcal{M}$ for natural numbers s.t. $\models^{\mathcal{M}}$ PA $+ \neg \forall n \forall x A[n, x]$.

Incompleteness theorem of Gödel

To every **consistent** extension T of PA in the same language there is a sentence A of PA such that $\models^{\mathcal{N}} T + A$ but neither $T \vdash A$ nor $T \vdash \neg A$.
Thus **arithmetic** is **essentially incomplete**, i.e. to every such T there is a **non-standard model of arithmetic** $\mathcal{M}$ such that $\models^{\mathcal{M}} T + \neg A$.

Extensions of PA

Lecture 7

Case and induction commands of CL

PA proves $x = 0 \lor \exists y\, x = y'$
this **justifies** a **case analysis** command *case* $N_1; z$ which behaves as:

$$\frac{}{z = 0 \mid z = y'}$$

Typically used when in assumptions $z \neq 0$ and we wish its predecessor.
PA proves $x + 1 = x'$ and, and hence $x = 0 \lor \exists y\, x = y + 1$
this **justifies** a **case analysis** command *case* $N; z$ which behaves as:

$$\frac{}{z = 0 \mid z = y + 1}$$

We also have N-induction rule: *ind* $N; x;$

$$\frac{}{\phi[0]* \;\Big|\; \begin{matrix}\phi[x] \\ \phi[x+1]*\end{matrix}}$$

Definitional extensions with predicate symbols

$$x < y \leftrightarrow \exists z\, x + z' = y$$
$$x \leq y \leftrightarrow x < y \lor x = y$$

PA now proves

- both symbols **transitive**,
- $<$ **irreflexive**: $x \not< x$,
- $\leq$ **antisymmetric**: $x \leq y \land y \leq x \rightarrow x = y$
- $<$ **linear**: $x < y \lor x = y \lor y < x$,
- $\leq$ **linear**: $x \leq y \lor y < x$

The last two **justify** the **case rules** of

Trichotomy $\mathit{case}\ \mathit{Trich}; x, y \ \dfrac{}{x < y \mid x = y \mid x > y}$

Dichotomy $\mathit{case}\ \mathit{Dich}; x, y \ \dfrac{}{x \leq y \mid x > y}$

Induction with measure

For any formula $\phi[\vec{x}]$ and term $\mu[\vec{x}]$ PA proves the **induction with measure**

$$\forall\vec{x}(\forall\vec{y}(\mu[\vec{y}] < \mu[\vec{x}] \to \phi[\vec{y}]) \to \phi[\vec{x}]) \to \phi[\vec{x}]$$

This **justifies** the induction **rule**: *indm* $\mu[\vec{x}]$

$$\frac{}{\dfrac{\forall\vec{y}(\mu[\vec{y}] < \mu[\vec{x}] \to \phi[\vec{y}])}{\phi[\vec{x}]*}}$$

A **special** case when the measure $\mu[\vec{x}]$ is just x we have **complete induction**:

$$\forall x(\forall y(y < x \to \phi[y]) \to \phi[x]) \to \phi[x]$$

and the **rule**: *indm* x

$$\frac{}{\dfrac{\forall y(y < x \to \phi[y])}{\phi[x]*}}$$

Definitional extensions with function symbols

In arbtritrary theory T a new f by its defining axiom:

$$f(\vec{x}) = y \leftrightarrow \phi[\vec{x}, y]$$

provided T proves the **existence** and **uniqueness** conditions:

$$\forall \vec{x} \exists y \phi[\vec{x}, y]$$

$$\phi[\vec{x}, y_1] \land \phi[\vec{x}, y_2] \rightarrow y_1 = y_2$$

If PA proves the **existence** $\forall \vec{x} \exists y \phi[\vec{x}, y]$ then we can **uniquely** pick the **least number** y **such that** $\phi[\vec{x}, y]$.
Extension of PA by **minimization** adds two new axioms

$$\phi[\vec{x}, f(\vec{x})] \qquad z < f(\vec{x}) \rightarrow \neg\phi[\vec{x}, z]$$

This is done by **defining** $f(\vec{x}) = \mu_y[\phi[\vec{x}, y]]$.
CL command *use* f makes the two axioms accessible.
Minimization is **equivalent** to extending PA with

$$f(\vec{x}) = y \leftrightarrow \phi[\vec{x}, y] \land \forall z(z < y \rightarrow \neg\phi[\vec{x}, z])$$

where both the **existence** and **uniqueness** are provable.

Introduction of modified subtraction

PA proves $\forall x \forall y \exists d (y \leq x \rightarrow y + d = x)$
We can thus extend PA by minimization:

$$x \dot{-} y = \mu_d [y \leq x \rightarrow y + d = x]$$

and the two axioms are accessible as

$$y \leq x \rightarrow y + (x \dot{-} y) = x$$
$$z < x \dot{-} y \rightarrow \neg (y \leq x \rightarrow y + z = x)$$

From the last we get $x < y \rightarrow x \dot{-} y \leq z$ and then

$$x < y \rightarrow x \dot{-} y = 0$$

Introduction of division and remainder functions

PA proves

$$\forall x \forall y \exists q \exists r (y > 0 \rightarrow x = q{\cdot}y + r \land r < y)$$

This is the existence condition for two extensions by minimizations:

$$x \div y = \mu_q[y > 0 \rightarrow \exists r(x = q{\cdot}y + r \land r < y)]$$
$$x \bmod y = \mu_r[y > 0 \rightarrow \exists q(x = q{\cdot}y + r \land r < y)]$$

Towards the recursive CL definitions

In order to be able to introduce functions defined in CL by recursion we need some kind of **coding** of sequences, i.e. **lists** in PA.
Cantor's pairing function does not suffice, we need also **list concatenation**!
We can introduce a **dyadic pairing** $x; y$ **and concatenation** $x \boxplus y$ functions s.t

$$x_1 ; y_1 = x_2 ; y_2 \rightarrow x_1 = x_2 \wedge y_1 = y_2$$
$$x < x; y \wedge y < x; y$$

Defining: $Atom(x) \leftrightarrow \forall y \forall z\, x \neq y; z$
we can introduce $\boxplus$ to satisfy:

$$Atom(x) \rightarrow x \boxplus z = z$$
$$(x; y) \boxplus z = x; y \boxplus z \ .$$

Predicates of divisibility and powers of two

We explicitly define the predicate of **divisibility** $x \mid y$:

$$x \mid y \leftrightarrow \exists z\, y = z{\cdot}x$$

and then the predicate $Pow_2(p)$ holding iff $p = 2^x$ for some x. In absence of exponentiation (it has a recursive definition) we can define the predicate of p **is a power of two** explicitly as:

$$Pow_2(p) \leftrightarrow \forall d(d \mid p \rightarrow d = 1 \lor 2 \mid d)$$

Introduction of dyadic concatenation into PA

Lecture 8

Review: Powers of two

Definition:

$$Pow_2(p) \leftrightarrow \forall d(d \mid p \rightarrow d = 1 \lor 2 \mid d)$$

Provably equivalent properties:

$$\neg Pow_2(0)$$
$$Pow_2(x\mathbf{1}) \leftrightarrow x = 0$$
$$Pow_2(x\mathbf{0}) \leftrightarrow x > 0 \land Pow_2(x)$$

This is equivalent again to **recursive clauses**:

$$Pow_2(x\mathbf{1}) \leftarrow x = 0$$
$$Pow_2(x\mathbf{0}) \leftarrow x > 0 \land Pow_2(x)$$

CL requires a default clause explicit:

$$Pow_2(x\mathbf{0}) \leftarrow x = 0 \land 0 = 1$$

We can now clausally redefine Pow_2 and let CL make **use** commands automatically

Binary case and induction

Clauses for Pow_2 are by **binary discrimination**:

$$x = y\mathbf{0} \lor x = y\mathbf{1}$$

proved from the properties of **division**
This justifies **Binary case rule in CL**: case $Nb; x$:

$$\frac{}{\begin{matrix} x = y\mathbf{0} \\ y = 0 \end{matrix} \;\middle|\; \begin{matrix} x = y\mathbf{0} \\ y > 0 \end{matrix} \;\middle|\; x = y\mathbf{1}}$$

Complete induction proves the schema of **Binary induction**:

$$\phi[0] \land \forall x(x > 0 \land \phi[x] \to \phi[x\mathbf{0}]) \land \forall x(\phi[x] \to \phi[x\mathbf{1}]) \to \phi[x]$$

This justifies **Binary induction rule in CL**: ind $Nb; x$

$$\frac{}{\begin{matrix} x = 0 \\ \phi[x\mathbf{0}]* \end{matrix} \;\middle|\; \begin{matrix} x > 0 \\ \phi[x] \\ \phi[x\mathbf{0}]* \end{matrix} \;\middle|\; \begin{matrix} \phi[x] \\ \phi[x\mathbf{1}]* \end{matrix}}$$

Towards dyadic concatenation in PA

We wish PA to **prove** the following recurrences as **theorems**

$$x \star 0 = x$$
$$x \star y\mathbf{1} = (x \star y)\mathbf{1}$$
$$x \star y\mathbf{2} = (x \star y)\mathbf{2}$$

For that we need to define $\star$ explicitly:

$$x \star y = x \cdot 2^{|y|} + y$$

For that we need to introduce into PA the **dyadic length power** function: $Dlp(x) \equiv 2^{|x|}$.
Note that we cannot directly define: $|x|$ or 2^x, but we can $2^{|x|}$.

$2^{|x|}$: Illustration

For x such that $7 \leq x \leq 14$ we have

$$2^{|x|} = \begin{cases} \mathbf{0112} & \text{if } x = 7 \;\; = \mathbf{0111} \\ \mathbf{0112} & \text{if } x = 8 \;\; = \mathbf{0112} \\ \mathbf{0112} & \text{if } x = 9 \;\; = \mathbf{0121} \\ \mathbf{0112} & \text{if } x = 10 = \mathbf{0122} \\ \mathbf{0112} & \text{if } x = 11 = \mathbf{0211} \\ \mathbf{0112} & \text{if } x = 12 = \mathbf{0212} \\ \mathbf{0112} & \text{if } x = 13 = \mathbf{0221} \\ \mathbf{0112} & \text{if } x = 14 = \mathbf{0222} \end{cases}$$

Note: $y \star x = y{\cdot}8 + x = y{\cdot}2^3 + x = y{\cdot}2^{|x|} + x$
Also note idempotency: $2^{|2^{|8|}|} = 2^{|8|}$.

Introduction of $2^{|x|}$ into PA

By extension by definition:

$$2^{|x|} = p \leftrightarrow Pow_2(p) \land p \leq x + 1 < 2 \cdot p$$

because for $x > 0$ we have

$$(\overbrace{1 \cdots 1}^{|x|})_2 = 2^{|x|} - 1 \leq x < 2^{|x|+1} - 1 = (\overbrace{1 \cdots 1}^{|x|+1})_2$$

We extend CL by **minimization**:

$$2^{|x|} = \mu_p[Pow_2(p) \land x + 1 < 2 \cdot p]$$

We need to prove the **existence condition**:

$$\exists p(Pow_2(p) \land x + 1 < 2 \cdot p)$$

which says that powers of two are **unbounded**

Comparison of dyadic length

We cannot define in PA the **dyadic length** $|x|$ yet, but we can compare the dyadic length of two numbers:
The numbers x and y have the same dyadic length iff
$2^{|x|} = 2^{|y|}$
or
The number x has a shorter dyadic length than y iff $2^{|x|} < 2^{|y|}$
This is possible because 2^x is **injective**

Recursive clauses for $2^{|x|}$

After defining

$$2^{|x|} = \mu_p[Pow_2(p) \land x + 1 < 2{\cdot}p]$$

PA proves:

$$2^{|0|} = 1$$
$$2^{|x\mathbf{1}|} = 2{\cdot}2^{|x|}$$
$$2^{|x\mathbf{2}|} = 2{\cdot}2^{|x|}$$

because **intuitively** $2^{|x\mathbf{1}|} = 2^{|x|+1} = 2{\cdot}2^{|x|}$
The clauses are by **dyadic recursion**.

Dyadic case and induction

Clauses for $2^{|x|}$ are by **dyadic discrimination**:

$$x = 0 \lor x = y\mathbf{1} \lor x = y\mathbf{2}$$

proved by binary case analysis

This justifies **Dyadic case rule in CL**: case $N_2; x$:

$$\frac{}{x = 0 \mid x = y\mathbf{1} \mid x = y\mathbf{2}}$$

Complete induction proves the schema of **Dyadic induction**:

$$\phi[0] \land \forall x(\phi[x] \to \phi[x\mathbf{1}]) \land \forall x(\phi[x] \to \phi[x\mathbf{2}]) \to \phi[x]$$

This justifies **Dyadic induction rule in CL**: ind $N_2; x$

$$\frac{}{\phi[0]* \;\Big|\; \begin{array}{c}\phi[x]\\ \phi[x\mathbf{1}]*\end{array} \;\Big|\; \begin{array}{c}\phi[x]\\ \phi[x\mathbf{2}]*\end{array}}$$

Clauses for $\star$ as theorems of PA

We explicitly define $x \star y = x \cdot 2^{|y|} + y$ and prove as theorems the clauses for $\star$ by **dyadic recursion**:

$$x \star 0 = x$$
$$x \star y\mathbf{1} = (x \star y)\mathbf{1}$$
$$x \star y\mathbf{2} = (x \star y)\mathbf{2}$$

We can now properties of dyadic concatenation by **dyadic induction** with automatical uses of clauses.

Auxiliary predicate

We explicitly define

$$D\_two(m) \leftrightarrow \exists m_1 \exists m_2\, m = m_1 \mathbf{2} \star m_2$$

Note that

$$m = m_1 \mathbf{2} \star m_2 = (m_1 \star 0\mathbf{2}) \star m_2 = m_1 \star 2 \star m_2$$

And prove as **theorems** its clauses by **dyadic recursion**:

$$D\_two(m\mathbf{1}) \leftarrow D\_two(m)$$
$$D\_two(m\mathbf{2})\ .$$

The relevance of D_two

This predicate is D_two important because PA proves

$$\exists n(2^{|x+1|} = n \star 2 \land \neg D\_two(n))$$

i.e. $2^{|x+1|} = (1 \cdots 1)_2 \star 2$.
PA then proves the existence of **leading powers**:

$$D\_two(m) \to \exists x \exists m_1\, m = 2^{|x+1|} \star m_1$$

i.e. if m contains 2 then $m = (\overbrace{1 \cdots 1}^{n})_2 \star 2 \star m_1$ for some m_1, n.
PA also proves the existence of **trailing ones**

$$\exists m_1 \exists n(m = m_1 \mathbf{0} \star n \land \neg D\_two(n))$$

i.e. $m = m_1\mathbf{0} \star (\overbrace{1 \cdots 1}^{|n|})_2$ for some m_1, n

Introduction of dyadic pairing into PA

Lecture 9

Review: Dyadic concatenation

$$x \mid y \leftrightarrow \exists z\, x{\cdot}z = y$$
$$Pow_2(p) \leftrightarrow \forall d((d+2) \mid p \to 2 \mid d)$$
$$2^{|x|} = \mu_p[Pow_2(p) \land x+1 < 2{\cdot}p]$$
$$x \star y = x{\cdot}2^{|y|} + y$$

We have also defined

$$D\_two(m) \leftrightarrow \exists n_1 \exists n_2\, m = n_1 \star 2 \star n_2 \;,$$

i.e. $\neg D\_two(m)$ iff $m = 1 \star \cdots \star 1$.

However, we have for $x > 0$ $\quad 2^{|x|} = \overbrace{1 \star \cdots \star 1 \star 2}^{|x|}$

Also $2^{|0|} = 1$. Thus, for any x $\quad 2^{|x|} \dot{-} 1 = \overbrace{1 \star \cdots \star 1}^{|x|}$ i.e.

$$\neg D\_two(m) \leftrightarrow \exists x\, m+1 = 2^{|x|} \leftrightarrow \exists x\, m = 2^{|x|} \dot{-} 1$$

Marker sequences

To every number m there are unique numbers n and b as well as numbers $a_1, a_2, \ldots, a_n$ of unique dyadic length such that

$$m = \overbrace{\overbrace{1 \star \cdots \star 1 \star 2}^{|a_1+1|} \star \overbrace{1 \star \cdots \star 1 \star 2}^{|a_2+1|} \star \cdots \star \overbrace{1 \star \cdots \star 1 \star 2}^{|a_n+1|}}^{n} \star \overbrace{1 \star \cdots \star 1}^{|b|}$$

This can be also written as:

$$m \quad = 2^{|a_1+1|} \quad \star\, 2^{|a_2+1|} \quad \star\, \cdots \quad \star\, 2^{|a_{n+1}|} \quad \star\, (2^{|b|} - 1)$$

Note that that $x + 1$ and $2^{|x+1|}$ have the same **length** because of **idempotency**: $2^{|x+1|} = 2^{|2^{|x+1|}|}$.
To every m and w of the same **length** there are **unique** n, b, $a_1, a_2, \ldots, a_n$ s.t.:

$$w = (a_1 + 1) \star (a_2 + 1) \star \cdots \star (a_n + 1) \star b$$
$$m = 2^{|a_1+1|} \quad \star\, 2^{|a_1+1|} \quad \star \cdots \star\, 2^{|a_n+1|} \quad \star\, (2^{|b|} - 1)$$

Properties of squares

Square is an **injection**:

$$x^2 = y^2 \rightarrow x = y$$

and $\sqrt{2}$ is irrational: $y > 0 \rightarrow (\frac{x}{y})^2 \neq 2$ for $x, y \in \mathbb{Z}$. This is expressed in PA as

$$x^2 = 2{\cdot}y^2 \rightarrow x = 0 \wedge y = 0$$

Hence

$$i{\cdot}x^2 = j{\cdot}y^2 \wedge x > 0 \wedge 0 < i,j \leq 2 \rightarrow i = j \wedge x = y$$

This property is used in the **uniqueness** property on the following slide.

Two lemmas for Splits

Existence of splits

$$\exists w \exists i \exists m (t = w \star i \star m \land 2^{|w|} = 2^{|m|} \land 2^{|i|} \leq 2)$$

We can view this as every number t can be decomposed into two numbers of length almost $\sqrt{|t|}$: $t = \begin{array}{|c|c|}\hline w & i \\ \hline \multicolumn{2}{c}{\star} \\ \hline m & \\ \hline \end{array}$ and $i = 0, 1, 2$

Uniqueness of splits:

$$w_1 \star i_1 \star m_1 = w_2 \star i_2 \star m_2 \land 2^{|i_1|} \leq 2 \land 2^{|w_1|} = 2^{|m_1|} \land$$
$$2^{|i_2|} \leq 2 \land 2^{|w_2|} = 2^{|m_2|} \rightarrow w_1 = w_2 \land i_1 = i_2 \land m_1 = m_2$$

Definition of splits

$$t \doteq \begin{bmatrix} v \\ m \end{bmatrix} \leftrightarrow t = v \star m \land \exists w \exists i (v = w \star i \land 2^{|m|} = 2^{|w|} \land 2^{|i|} \leq 2)$$

We can vizualize the definition as: $t = \begin{array}{c} \begin{array}{|c|c|} \hline w & i \\ \hline \end{array} \\ \begin{array}{|c|} \hline \quad v \quad \\ \hline \end{array} \\ \star \\ \begin{array}{|c|c|} \hline m & \\ \hline \end{array} \end{array}$.

Splits **exist** $\exists v \exists m\, t \doteq \begin{bmatrix} v \\ m \end{bmatrix}$ and they are **unique**:

$$t \doteq \begin{bmatrix} v_1 \\ m_1 \end{bmatrix} \land t \doteq \begin{bmatrix} v_2 \\ m_2 \end{bmatrix} \to v_1 = v_2 \land m_1 = m_2$$

Adjustment of splits

If $t \doteq \begin{bmatrix} v \\ m \end{bmatrix}$ then $t = w \star i$ where $i = 0, 1, 2$ and

$$w = (a_1 + 1) \star (a_2 + 1) \star \cdots \star (a_n + 1) \star b$$

$$m = 2^{|a_1+1|} \star 2^{|a_2+1|} \star \cdots \star 2^{|a_n+1|} \star (2^{|b|} - 1)$$

The **tail** part such that: $b \star i \star (2^{|b|} - 1) \doteq \begin{bmatrix} b \star i \\ 2^{|b|} - 1 \end{bmatrix}$ is an **atom**.

We now define $Adj(t) \doteq \begin{bmatrix} w' \\ m' \end{bmatrix}$ removing the atom:

$$w' = (a_1 + 1) \star (a_2 + 1) \star \cdots \star (a_n + 1)$$

$$m' = 2^{|a_1+1|} \star 2^{|a_2+1|} \star \cdots \star 2^{|a_n+1|}$$

$$Adj(t) = s \leftrightarrow \exists v_1 \exists v_2 \exists m \exists b (t \doteq \begin{bmatrix} v_1 \star v_2 \\ m\mathbf{0} \star (2^{|b|} \dot{-} 1) \end{bmatrix} \land 2^{|v_1|} = 2^{|m\mathbf{0}|} \land s \doteq \begin{bmatrix} v_1 \\ m\mathbf{0} \end{bmatrix})$$

Note that: $v_1 = w'$, $m\mathbf{0} = m'$, and $v_2 = b \star i$.

Dyadic list concatenation and pairing

We define the **concatenation**:

$$s \boxplus t = u \leftrightarrow \exists v_1 \exists v_2 \exists m \exists x (Adj(s) \doteq \begin{bmatrix} v_1 \\ m_1 \end{bmatrix} \wedge t \doteq \begin{bmatrix} v_2 \\ m_2 \end{bmatrix} \wedge u \doteq \begin{bmatrix} v_1 \star v_2 \\ m_1 \star m_2 \end{bmatrix})$$

and the **pairing**:

$$x; t = ((x+1) \star 2^{|x+1|}) \boxplus t$$

Thus

$$t \doteq \begin{bmatrix} v \\ m \end{bmatrix} \rightarrow s; t \doteq \begin{bmatrix} (x+1) \star v \\ 2^{|x+1|} \star m \end{bmatrix}$$

Nested iteration

Lecture 10

The schema of nested iteration

For every three-place function g, unary **measure** function μ, and a constant C giving a **recursion count** introduced into PA such that

$$\vdash g(x,n,a) = v\mathbf{1} \rightarrow \mu(v) < \mu(x)$$
$$\vdash 2 \mid g(x,0,a)$$

we wish to introduce a three-place **nested iteration** function g^* such that:

$$\vdash g^*(x,n,a) = y \leftarrow g(x,n,a) = y\mathbf{0}$$
$$\vdash g^*(x,n,a) = y \leftarrow g(x,n,a) = v\mathbf{1} \land n = m+1 \land g^*(v,C,0) = w \land g^*(x,m,a \boxplus (w;0)) = y$$

The measure of this recursion is $\mu(x){\cdot}C + n$ because $\mu(x){\cdot}C + n > \mu(x){\cdot}C + m$ (for the outer recursion) and $\mu(x){\cdot}C + n > (\mu(v)+1){\cdot}C = \mu(v){\cdot}C + C$ (for the inner recursion).

Example: Reduction of Fibonacci sequence to nested iteration

$F_0 = F_1 = 1$ and $F_{x+2} = F_x + F_{x+1}$. For this **explicitly** define $C = 2$, $\mu(x) = x$, and

$$g(x,n,a) = \begin{cases} (x \dot{-} 2)\mathbf{1} & \text{if } x \geq 2 \land n \geq 2 \\ (x \dot{-} 1)\mathbf{1} & \text{if } x \geq 2 \land n = 1 \land a = u; b \\ (u+z)\mathbf{0} & \text{if } x \geq 2 \land a = u; z; b \\ 1\mathbf{0} & \text{otherwise} \end{cases}$$

Since PA proves $2 \mid g(x,0,a)$ and $g(x,n,a) = v\mathbf{1} \rightarrow v < x$ we can use the schema of iteration:

$$\vdash g^*(x,n,a) = v \leftarrow g(x,n,a) = v\mathbf{0}$$
$$\vdash g^*(x,n,a) = y \leftarrow g(x,n,a) = v\mathbf{1} \land n = m+1 \land g^*(v,C,0) = w \land g^*(x,m,a \boxplus (w;0)) = y$$

We can now explicitly define $F_x = g^*(x,C,0)$ and prove in PA the **recurrences** for F.

Arithmetization of computation trees for g^*

We will code derivations of identities $g^*(\underline{x}, \underline{n}, \underline{a}) = \underline{y}$ where $\underline{i}$ abbreviates $S^i(0)$.

The nodes in trees satisfy **local conditions**:

$$\frac{g^*(\underline{x}, \underline{n}, \underline{a}) = \underline{y}}{0 \mid 0} \qquad \text{if } g(x, n, a) = y\mathbf{0}$$

$$\frac{g^*(\underline{x}, \underline{n+1}, \underline{a}) = \underline{y}}{g^*(\underline{v}, C, 0) = \underline{w} \mid g^*(\underline{x}, \underline{n}, \underline{a \boxplus (w;0)}) = \underline{y}} \qquad \text{if } g(x, n+1, a) = v\mathbf{1}$$

We **arithmetize** $g^*(x, n, a) = y$ as $Lb(x, n, a, y)$ where $Lb(x, n, a, y) = x; n; a; y$ and abbreviate this to $(\mathbf{g}^*(x, n, a) \stackrel{\bullet}{=} y)$.

The predicate Ct

Computation trees are **flattened** to lists containing $(\mathbf{g}^*(x,n,a)\dot{=}^\bullet y)$ such that for $t = (\mathbf{g}^*(x,n,a)\dot{=}^\bullet y); s$ the list s contains the sons (if any).

$$
\begin{aligned}
&Lcond(x,n,a,y,t) \leftrightarrow \exists v(g(x,n,a) = v\mathbf{0} \wedge v = y \vee \\
&\quad \exists m \exists w(n = m+1 \wedge g(x,n,a) = v\mathbf{1} \wedge \\
&\qquad (\mathbf{g}^*(v,C,0)=^\bullet w)\ \varepsilon\ t \wedge (\mathbf{g}^*(x,m,a \boxplus (w;0))=^\bullet y)\ \varepsilon\ t))
\end{aligned}
$$

$$Ct(s) \leftrightarrow \forall x \forall n \forall a \forall y \forall t((\mathbf{g}^*(x,n,a)=^\bullet y); t \sqsubseteq s \rightarrow Lcond(x,n,a,y,t))$$

We then prove

$$Ct(s) \wedge t \sqsubseteq s \rightarrow Ct(t)$$

$$Adj(s) = 0 \rightarrow Ct(s)$$

$$\forall x \forall n \forall a \forall y\ b \neq (\mathbf{g}^*(x,n,a)=^\bullet y) \rightarrow Ct(b;s) \leftrightarrow Ct(s)$$

$$Ct((\mathbf{g}^*(x,n,a)=^\bullet y);s) \leftrightarrow Lcond((x,n,a,y,s) \wedge Ct(s)$$

$$Ct(s) \wedge Ct(t) \rightarrow Ct(s \boxplus t)$$

Graph of nested iteration function

We introduce a four-place predicate $\mathbf{g}^*(x,n,a) \doteq y$, which will be the **graph** of g^*:

$$\mathbf{g}^*(x,n,a) \doteq y \leftrightarrow \exists t\, Ct((\mathbf{g}^*(x,n,a) =^{\bullet} y); t) \ . \tag{1}$$

We have the following **recurrences**:

$$\vdash g(x,n,a) = v\mathbf{0} \rightarrow \mathbf{g}^*(x,n,a) \doteq y \leftrightarrow y = v$$

$$\vdash g(x,n+1,a) = v\mathbf{1} \rightarrow \mathbf{g}^*(x,n+1,a) \doteq y \leftrightarrow$$
$$\exists w(\mathbf{g}^*(v,C,0) \doteq w \wedge \mathbf{g}^*(x,n,a \boxplus (w;0)) \doteq y)$$

Introducttion of nested iteration function

By **measure induction** with $\mu(x) \cdot C + n$ we prove the **existence** and **uniqueness** which assert that $\mathbf{g}^*(x, n, a) \doteq y$ is indeed a **graph**:

$$\vdash \exists y \mathbf{g}^*(x, n, a) \doteq y$$
$$\vdash \mathbf{g}^*(x, n, a) \doteq y_1 \land \mathbf{g}^*(x, n, a) \doteq y_2 \to y_1 = y_2$$

We can thus introduce g^* by **minimization**:

$$g^*(x, n, a) = \mu_y[\mathbf{g}^*(x, n, a) \doteq y]$$

and prove the desired **recurrences**.

Explicit clausal definitions

Lecture 11

Examples of Discriminators built into CL

Discriminators **without patterns**:

- **negation**: $\mathbf{A} \mid \neg\mathbf{A}$
- **test on zero**: $\mathbf{s} = 0 \mid \mathbf{s} > 0$
- **trichotomy**: $\mathbf{s} < \mathbf{t} \mid \mathbf{s} = \mathbf{t} \mid \mathbf{s} > \mathbf{t}$

Discriminators **with patterns**:

- **let**: $\mathbf{s} = z$
- **binary**: $\mathbf{s} = z\mathbf{0} \wedge z = 0 \mid \mathbf{s} = z\mathbf{0} \wedge z > 0 \mid \mathbf{s} = z\mathbf{1}$
- **division by four**: $\mathbf{s} = 4{\cdot}z + v \wedge 0 \leq v \leq 3$

- **exactly** one alternative holds
- pattern variables **uniquely exist**

Examples of Provable discriminators

- discr. on the **head of lists:** $Adj(\mathbf{s}) = 0 \mid \mathbf{s} = z; t$
- discr. on the **tail of lists:** $Adj(\mathbf{s}) = 0 \mid \mathbf{s} = t \boxplus (z; u) \land Adj(u) = 0$
-

The head discrimination used in a clausal definition:

$$Rev(t) = t \qquad \leftarrow Adj(t)$$
$$Rev(x; t) = Rev(t) \boxplus (x; 0)$$

Conditional discriminators:

general division: provided $\mathbf{t} > 0$ then $\mathbf{s} = \mathbf{t}\cdot z + v \wedge 0 \leq v < \mathbf{t}$
special discrimination for g^*: provided PA proves

$$g(x,n,a) = v\mathbf{1} \rightarrow \mu(v) < \mu(x)$$
$$2 \mid g(x,0,a)$$

we have $g(\mathbf{s},\mathbf{n},\mathbf{a}) = v\mathbf{0} \mid g(\mathbf{s},\mathbf{n},\mathbf{a}) = v\mathbf{1} \wedge \mathbf{n} = m+1$
This is used in the clauses for g^*:

$$g^*(x,n,a) = v \qquad \leftarrow g(x,n,a) = v\mathbf{0}$$
$$g^*(x,n+1,a) = g^*(x,n,a \boxplus (g^*(v,C,0);0)) \leftarrow g(x,n+1,a) = v\mathbf{1}$$

General form of provable discriminators

We use **bold** variables $\mathbf{x}$ for possibly empty sequences of variables $x_1, \ldots, x_n$,
we let $\exists\mathbf{x}\mathbf{D}$ to stand for $\exists x_1 \ldots \exists x_n \mathbf{D}$ (n can be empty), and write $\mathbf{x} = \mathbf{y}$ for $x_1 = y_1 \wedge \cdots x_n = y_n$.
Suppose that PA proves for $k \geq 1$:

$$\exists\mathbf{z}_1\mathbf{D}_1[\mathbf{z}_1] \vee \exists\mathbf{z}_2\mathbf{D}_2[\mathbf{z}_2] \vee \cdots \vee \exists\mathbf{z}_k\mathbf{D}_k[\mathbf{z}_k]$$

$$\mathbf{D}_i[\mathbf{z}_i] \rightarrow \neg\mathbf{D}_j[\mathbf{z}_j] \qquad \text{for all } 1 \leq i \neq j \leq k$$

$$\mathbf{D}_i[\mathbf{z}_i] \wedge \mathbf{D}_i[\mathbf{w}] \rightarrow \mathbf{z}_i = \mathbf{w} \qquad \text{for all } 1 \leq i \leq k$$

This means that **exactly** one $\mathbf{D}_i[\mathbf{z}]_i$ holds with **uniquely** determined patterns $\mathbf{z}_i$.
We can then use

$$\mathbf{D}_1[\mathbf{z}_1] \mid \mathbf{D}_2[\mathbf{z}_2] \mid \cdots \mid \mathbf{D}_k[\mathbf{z}_k]$$

as **provable discriminators** (we can even permit conditional discrimination).

Clausal formulas

In the following we will write $\mathbf{A}[\mathbf{x}; v]$ for a formula with the **output** variable v free and with other free variables among the **input** variables $\mathbf{x}$

A formula $\mathbf{A}[\mathbf{x}; v]$ is a **clausal formula** if $\mathbf{A}$ is either of a form

- $\mathbf{s}[\mathbf{x}] = v$ or
- $\exists \mathbf{z}_1(\mathbf{D}_1[\mathbf{x}, \mathbf{z}_1] \wedge \mathbf{A}_1[\mathbf{x}, \mathbf{z}_1; v]) \vee \cdots \vee \exists \mathbf{z}_k(\mathbf{D}_k[\mathbf{x}, \mathbf{z}_k] \wedge \mathbf{A}_k[\mathbf{x}, \mathbf{z}_k, v])$

where $\mathbf{D}_1, \ldots, \mathbf{D}_k$ is a provable discriminator and $\mathbf{A}_1, \ldots, \mathbf{A}_k$ are clausal formulas.

Using clausal formulas in explicit definitions

In all clausal formulas $\mathbf{A}[\mathbf{x}; v]$ for every $\mathbf{x}$ the output variable is uniquely determined, i.e. PA proves:

$$\exists v \mathbf{A}[\mathbf{x}; v]$$

$$\mathbf{A}[\mathbf{x}; v] \land \mathbf{A}[\mathbf{x}; w] \rightarrow v = w$$

We can thus explictly introduce into PA a new function symbol f by:

$$f(\mathbf{x}) = v \leftrightarrow \mathbf{A}[\mathbf{x}; v] \ ,$$

or in CL by $f(\mathbf{x}) = \mu_v[\mathbf{A}[\mathbf{x}; v]]$.
The above equivalence is actually equivalent in PA to

$$f(\mathbf{x}) = v \leftarrow \mathbf{A}[\mathbf{x}; v]$$

because if in the direction $(\rightarrow)$ $f(\mathbf{x}) = v$ holds then $\mathbf{A}[\mathbf{x}; w]$ for some w by existence and $w = f(\mathbf{x})$ by $(\leftarrow)$.

Unfolding the clausal formulas

We now assign to every formula **B**, every clausal formula $\mathbf{A}[\mathbf{x}; v]$ and a new function symbol f a **finite set** of **clauses** by the **unfolding** operator $U[f, \mathbf{B}, \mathbf{A}]$ such that:
• if $\mathbf{A} \equiv \mathbf{s}[\mathbf{x}] = v$ then $U[f, \mathbf{B}, \mathbf{A}] = \{f(\mathbf{x}) = v \leftarrow \mathbf{B} \wedge \mathbf{s}[\mathbf{x}] = v\}$ and if

$$\mathbf{A} \equiv \exists \mathbf{z}_1(\mathbf{D}_1[\mathbf{x}, \mathbf{z}_1] \wedge \mathbf{A}_1[\mathbf{x}, \mathbf{z}_1; v]) \vee \cdots \vee \exists \mathbf{z}_k(\mathbf{D}_k[\mathbf{x}, \mathbf{z}_k] \wedge \mathbf{A}_k[\mathbf{x}, \mathbf{z}_k, v])$$

then

$$U[f, \mathbf{B}, \mathbf{A}] = \cup_{1 \leq i \leq k} U[f, (\mathbf{B} \wedge \mathbf{D}_i[\mathbf{x}, \mathbf{z}_i]), \mathbf{A}_i[\mathbf{x}, \mathbf{z}_i, v]]$$

If $U[f, \top, \mathbf{A}[\mathbf{x}; v]] = \{\mathbf{C}_1, , \ldots \mathbf{C}_m\}$ then we have

$$\vdash f(\mathbf{x}) = b \leftarrow \mathbf{A}[\mathbf{x}; v] \text{ iff } \vdash \mathbf{C}_1 \text{ and } \ldots \text{and } \vdash \mathbf{C}_m$$

Recursive clausal definitions

Lecture 12

Recursive clausal formulas

For recursive clausal definitions we extend **clausal formulas for** f to **recursive** ones with a new rule:

- $f(\mathbf{s}[\mathbf{x}]) = z \wedge \mathbf{A}_1[\mathbf{x}, z; v]$ is a recursive clausal formula if
  - $\mathbf{s}$ is a sequence of terms not applying the function symbol f and,
  - $\mathbf{A}_1[\mathbf{x}, z; v]$, is a recursive clausal formula.

Our plan is to extend PA for **suitable** clausal formulas $\mathbf{A}[\mathbf{x}; v]$ by extension by definition of f such that

$$\vdash f(\mathbf{x}) = v \leftarrow \mathbf{A}[\mathbf{x}; v]$$

and then **unfold** this into provably equivalent **recursive clauses** for f

Iterated functions $g_{\mathbf{A}}$

For every **recursive** clausal formula $\mathbf{A}[\mathbf{x}; v]$ we will define an **explicit** clausal formula $\mathbf{B}[n, a, \mathbf{x}; v]$ for an explicit definition of a three-argument function $g_{\mathbf{A}}(x, n, a)$ (below only g) such that

$$\vdash g((x_1; \ldots; x_n), n, a) = v \leftarrow \mathbf{B}[n, a, \mathbf{x}; v]$$

(when x is not an n-tuple then g yields 0) and for an unary measure function μ and a numeral $C \equiv \underline{k}$ we have

$$\vdash g(x, n, a) = v\mathbf{1} \rightarrow \mu(v) < \mu(x)$$
$$\vdash 2 \mid g(x, 0, a)$$

Such an **A** is called **regular**.
We then define the **iteration** function g^* and from it explicitly

$$f(\mathbf{x}) = g^*((x_1; \ldots; x_n), C, 0)$$

PA will then prove the **recursive clauses unfolded** from **A**.

Construction of B

By recursion on the structure of **A**. When **A** is:

- $\mathbf{s}[\mathbf{x}] = v$ then $\mathbf{B}[n, a, \mathbf{x}]$ is $(\mathbf{s}[\mathbf{x}])\mathbf{0} = v$.
- $\exists \mathbf{z}_1(\mathbf{D}_1[\mathbf{x}, \mathbf{z}_1] \wedge \mathbf{A}_1[\mathbf{x}, \mathbf{z}_1; v]) \vee \cdots \vee \exists \mathbf{z}_k(\mathbf{D}_k[\mathbf{x}, \mathbf{z}_k] \wedge \mathbf{A}_k[\mathbf{x}, \mathbf{z}_k, v])$ then **B** is $\exists \mathbf{z}_1(\mathbf{D}_1[\mathbf{x}, \mathbf{z}_1] \wedge \mathbf{B}_1[n, a, \mathbf{x}, \mathbf{z}_1; v]) \vee \cdots \vee \exists \mathbf{z}_k(\mathbf{D}_k[\mathbf{x}, \mathbf{z}_k] \wedge \mathbf{B}_k[n, a, \mathbf{x}, \mathbf{z}_k, v])$
- $f(\mathbf{s}[\mathbf{x}]) = z \wedge \mathbf{A}_1[\mathbf{x}, z; v]$ we obtain $\mathbf{B}_1[n, a, \mathbf{x}, z; v]$ by IH and set **B** to

$$Adj(a) = 0 \wedge (n = 0 \wedge (0)\mathbf{0} = v \vee n > 0 \wedge (\mathbf{s}[\mathbf{x}])\mathbf{1} = v) \vee \exists z \exists b(a = z; b \wedge \mathbf{B}_1[m, b, \mathbf{x}, z; v]))$$

Clausal definitions of predicates P

are by **clausal** definitions of their **characteristic** functions f such that $\vdash f(\mathbf{x}) = v \leftarrow \mathbf{A}[\mathbf{x}; v]$ where the (recursive) clausal formula $\mathbf{A}$ has the final **assignments** of the form $1 = v$ (true) or $0 = v$ (false) and **recursions** in it are always followed by **discriminations** on zero:

$$f(\mathbf{s}) = z \land (z = 0 \land \mathbf{A}_1 \lor z > 0 \land \mathbf{A}_1)$$

where neither $\mathbf{A}_1$ nor $\mathbf{A}_2$ contain z free.
We then explicitly **define** $P(\mathbf{x}) \leftrightarrow f(\mathbf{x}) > 0$ and prove in PA the (recursive) clauses for P obtained by unfolding of $\mathbf{A}$ where:

$$f(\mathbf{x}) = v \leftarrow \mathbf{B} \land 1 = v \quad \Rightarrow \quad P(\mathbf{x}) \leftarrow \mathbf{B}$$
$$f(\mathbf{x}) = v \leftarrow \mathbf{B} \land 0 = v \quad \Rightarrow \quad \neg P(\mathbf{x}) \leftarrow \mathbf{B}$$

We also change all above unfolded recursions as follows:

$$[\neg]P(\mathbf{x}) \leftarrow \mathbf{B} \land f(\mathbf{s}){=}z \land z{=}0 \land \mathbf{A}_1 \Rightarrow [\neg]P(\mathbf{x}) \leftarrow \cdots \neg P(\mathbf{s}) \land \mathbf{A}_1$$
$$[\neg]P(\mathbf{x}) \leftarrow \mathbf{B} \land f(\mathbf{s}){=}z \land z{>}0 \land \mathbf{A}_2 \Rightarrow [\neg]P(\mathbf{x}) \leftarrow \cdots P(\mathbf{s}) \land \mathbf{A}_2$$