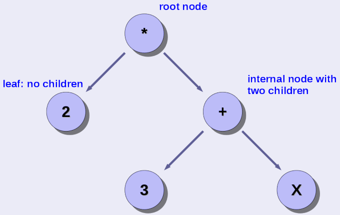
Part 2
Logic programs as inference systems

Terms

Concrete and abstract syntax

Let us consider the expression: $2 * (3 + X)$

- concrete syntax: the expression is a string made of seven characters (which must obey some well-formedness rule)
- abstract syntax: emphasis on the inherent hierarchical structure. $2 * (3 + X)$ corresponds to a tree:



Standard textual representation:
$*(2, +(3, X))$, or $times(2, plus(3, X))$ if we prefer names over special characters.

Functors, arities and variables

Elementary blocks

Terms are built on top of

- operation names/symbols (called functors in Prolog)
- logical variables (or simply variables); we use the standard Prolog convention: variables begin with an upper case letter

Arity

Every functor is associated with a fixed and finite arity

- $plus/2$ means *plus* has arity 2 (it takes two arguments)
- $plus/1$, $plus/2$: two distinct functors with the same name, but different arities
- 3 is a functor with arity 0, that is, a constant

Terms and trees

Well-formedness rules

Terms are trees where nodes are labeled by functors or variables

- standard rules on well-formed trees (see next slides)
- a node labeled by a functor of arity n must have exactly n children. Consequence: constants can only label leaves
- variables can only label leaves

Ground terms

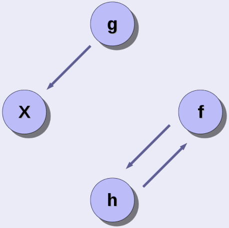
A term is ground if it contains no variables

More on trees (1)

Standard rules

1. There exists a unique node, called root, with no parents
2. All other nodes have exactly one parent
3. The ancestor/descendant relation cannot be cyclic

The ancestor/descendant relation is the transitive closure of the parent/child relation.
This is not a tree (rule 3 does not hold)



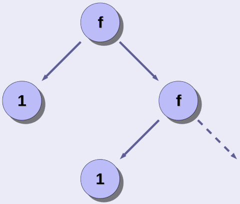
More on trees (2)

Finite and infinite trees

We use finite /infinite trees for representing terms and proofs (see later).

- Branching is always finite
- Depth is allowed to be infinite

Example:



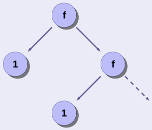
This tree represents the infinite term $f(1, f(1, f(1, f(\ldots$

Regular trees

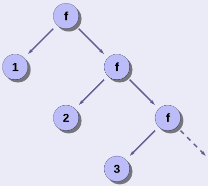
Definition

A tree is regular if it represents a term that has a finite number of subterms.
Equivalent terminology: rational tree (we will discover why) Examples:

- All finite trees are regular
- The following infinite tree is regular:



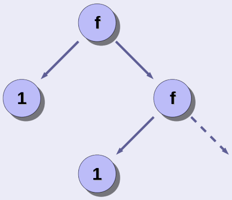
- The following tree is not regular:

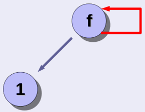


Representing infinite regular trees

Infinite regular trees as graphs

Graphs generalize trees (trees are particular kinds of graphs)
An infinite regular tree and its finite representation as a graph





Intuition: the graph infinitely unfolds to the regular tree

Substitutions

Definition

A substitution is a finite mapping from variables to terms.
Example: $\sigma = [X \mapsto f(1), Y \mapsto X]$
Application of σ to terms:

- $X\sigma = f(1)$
- $g(X, Y)\sigma = g(f(1), X)$ (variables are substituted in parallel)
- $f(Z)\sigma = f(Z)$ (variables for which no substitution is specified are implicitly mapped to themselves)

Composition of substitutions

Composition of substitutions σ_1 and σ_2:
the map $\sigma = \sigma_1\sigma_2$ s.t. $t\sigma = (t\sigma_1)\sigma_2$ for all terms t

Grounding substitution

A substitution σ is grounding for a term t if $t\sigma$ is a ground term

Matching and unification

Matching (functional programming)

A term t_1 matches a term t_2 if there exists a substitution σ s.t. $t_1 = t_2\sigma$.
Usually t_1 is ground and t_2 is not, and t_2 does not contain distinct variables (even though these conditions are not strictly necessary).
Examples:

- $f(1, g(2))$ matches $f(X, g(Y))$ with substitution $[X \mapsto 1, Y \mapsto 2]$
- $f(1, 2)$ does not match $f(X, g(Y))$

Unification (logic programming)

Terms t_1 and t_2 unify if there exists a substitution σ s.t. $t_1\sigma = t_2\sigma$.
Examples:

- $f(X, g(f(Z)))$ and $f(1, g(Y))$ unify with substitution $[X \mapsto 1, Y \mapsto f(Z)]$
- $f(X, 2)$ and $f(1, X)$ do not unify

Matching is unidirectional, unification is bidirectional

More on unification

Most general unifier

The following are all valid unifiers for $f(X, g(f(Z)))$ and $f(1, g(Y))$

1. $[X \mapsto 1, Y \mapsto f(Z)]$ (most general)
2. $[X \mapsto 1, Y \mapsto f(Z), W \mapsto a]$
3. $[X \mapsto 1, Y \mapsto f(0), Z \mapsto 0]$

Substitution 1 specifies the minimal set of equations between variables and terms needed to ensure unification

If two terms are unifiable, then there is always a most general unifier

Unification with regular terms: do $f(1, X)$ and X unify?

- No, if X can only be substituted with finite terms: if X occurs in a term t, and $t \neq X$, then X and t do not unify (occurs check)
- Yes, if X can be substituted with regular terms: $[X \mapsto f(1, f(1, f(1, f(\ldots))))]$

$f(1, f(1, f(1, f(\ldots))))$ is the unique solution of the syntactic equation $X = f(1, X)$

Herbrand universe and base

Herbrand universe (*HU*)

Let S be a finite set (called signature) of functor names with their arities

- inductive *HU* over S: all finite ground terms built on S
- coinductive HU^{co} over S: all finite and infinite ground terms built on S

Example: $S = \{z/0, s/1\}$

- inductive *HU* over S: $z, s(z), s(s(z)), \ldots$
- coinductive HU^{co} over S: inductive *HU* plus $s(s(s(\ldots$

Atoms and Herbrand base (*HB*)

An atom: $p(t_1, \ldots, t_n)$, where p is a predicate symbol of arity n (written p/n), and $t_1, \ldots, t_n$ are n terms.
An atom is ground, when all terms $t_1, \ldots, t_n$ are ground.
Example of atoms: $is\_nat(s(z)), odd(s(X)), geq(s(s(Y)), s(X))$

- inductive *HB*: all finite ground atoms
- coinductive HB^{co}: all finite and infinite ground atoms

Herbrand interpretation of predicate symbols

Example

Let *geq*/2 be a predicate symbol.
The interpretation of *geq*/2: a predicate, that is, a function taking two ground terms and returning either *false* or *true*
Predicate symbol interpretation as sets of ground atoms: all and only all ground atoms that are true

$$\{geq(z, z), geq(s(z), z), geq(s(z), s(z)), \ldots\}$$

A predicate symbol interpretation is a subset of *HB*

In fact, the interpretation of a predicate is a set of tuples, that is, a relation.

Definite Horn clauses

Definite Horn clauses (or simply Horn clauses) are meta-rules

Example

$$\frac{geq(X, Y), geq(Y, X)}{eq(X, Y)} \qquad \frac{premises}{conclusion}$$

Intended meaning:

if $geq(X, Y)$ and $geq(Y, X)$ hold, then $eq(X, Y)$ holds as well.

Prolog notation: $eq(X, Y)$:- $geq(X, Y), geq(Y, X)$.

Prolog terminology: *head* :- *body*

Facts

A fact is a meta-rule with no premises (an axiom) (or a Horn clause with an empty body). Example:

$$\frac{}{is\_nat(z)}$$

Intended meaning: $is\_nat(z)$ holds

Prolog notation: $is\_nat(z)$.

Ground instantiations of Horn clauses

Ground instantiations of Horn clauses are rules obtained by applying a grounding substitution to a Horn clause (a meta-rule)

Example

$$\frac{geq(s(s(z)), s(z)), geq(s(z), s(s(z)))}{eq(s(s(z)), s(z))}$$

is a rule which is the ground instantiation of the meta-rule

$$\frac{geq(X, Y), geq(Y, X)}{eq(X, Y)}$$

obtained by applying the substitution $\{X \mapsto s(s(z)), Y \mapsto s(z)\}$

Logic programs as inference systems

Inference systems as logic programs: defined by functors, predicate symbols, and a collection of meta-rules (a logic program, using the Prolog terminology)

A simple example

Functors: $s/1$, $z/0$
Predicates: $is\_nat/1$
Horn clauses/meta-rules:

$$\frac{}{is\_nat(z)}$$

$$\frac{is\_nat(N)}{is\_nat(s(N))}$$

Logic programs as inference systems

Inference systems as logic programs: defined by functors, predicate symbols, and a collection of meta-rules (a logic program, using the Prolog terminology)

A simple example

Functors: $s/1$, $z/0$
Predicates: $is\_nat/1$
Horn clauses/meta-rules:
In Prolog notation

$$is\_nat(z).$$
$$is\_nat(s(N)) \text{ :- } is\_nat(N).$$

Logic programs as inference systems

Inference systems as logic programs: defined by functors, predicate symbols, and a collection of meta-rules (a logic program, using the Prolog terminology)

A simple example

Functors: $s/1$, $z/0$
Predicates: $is\_nat/1$
Horn clauses/meta-rules:
In Prolog notation

$$is\_nat(z).$$
$$is\_nat(s(N)) \text{ :- } is\_nat(N).$$

Interpretation of logic programs

How predicate *is_nat* is defined by the meta-rules above?
Two equivalent ways to define the abstract (or declarative) semantics of an inference system

1. based on fixed points
2. based on proof trees

Fixed point semantics

Fixed point of one step inference function

Example

Function directly defined in terms of the meta-rules
Intuition: $f(A)$ = all ground atoms that can be inferred in one step from A with the rules (= ground instantiations of the meta-rules)

$$f(A) = \{is\_nat(z)\} \cup \{is\_nat(s(t)) \mid is\_nat(t) \in A\}$$

Remarks:

- $is\_nat(z)$ is a fact, hence it can be inferred in one step from any set
- A is a set of ground atoms
- t is a ground term
- $\dfrac{is\_nat(t)}{is\_nat(s(t))}$ is a ground instantiation of $\dfrac{is\_nat(N)}{is\_nat(s(N))}$

One step inference

General definition

Given a set A of ground atoms, and the generic meta-rule R

$$\frac{p_1(\bar{t}_1), \ldots, p_n(\bar{t}_n)}{p_0(\bar{t}_0)}$$

where $\bar{t}_0, \ldots, \bar{t}_n$ are tuples of terms

$p_0(\bar{g}_0)$ can be inferred in one step from A with R iff

1. $\dfrac{p_1(\bar{g}_1), \ldots, p_n(\bar{g}_n)}{p_0(\bar{g}_0)}$ is a ground instantiation of R
2. and $\{p_1(\bar{g}_1), \ldots, p_n(\bar{g}_n)\} \subseteq A$

Remark: if R is an axiom, then 2 trivially holds since $\emptyset \subseteq A$

Inductive and coinductive interpretation with fixed points (1)

- One step inference is always a **monotone** function
- By the Tarski-Knaster theorem f has a least and a greatest fixed point
- Inductive interpretation: lfp f, $f : \wp(HB) \to \wp(HB)$
- Coinductive interpretation: gfp f, $f : \wp(HB^{co}) \to \wp(HB^{co})$
- One step inference is always a function f preserving sup of ascending chain $f^0(\emptyset) \subseteq \ldots \subseteq f^n(\emptyset) \subseteq$
- One step inference is a function f preserving inf of descending chain $f^0(U) \supseteq \ldots \supseteq f^n(U) \supseteq$ **only when** $U = HB^{co}$
- We can apply the Kleene theorem to compute the least and the greatest fixed point

Inductive and coinductive interpretation with fixed points (2)

Example with meta-rules for *is_nat*

$f(A) = \{is\_nat(z)\} \cup \{is\_nat(s(t)) \mid is\_nat(t) \in A\}$

$f(\emptyset) = \{is\_nat(z)\}$
$f^2(\emptyset) = f(\{is\_nat(z)\}) = \{is\_nat(z), is\_nat(s(z))\}$
...
$f^n(\emptyset) = \{is\_nat(z), is\_nat(s(z)), \ldots, is\_nat(s^n(z))\}$ ($s^n(z)$ = s applied to z n times)
$\text{lfp } f = \{is\_nat(s^n(z)) \mid n \in \mathbb{N}\}$

$f(HB^{co}) = \{is\_nat(s^n(z)) \mid n \in \mathbb{N}\} \cup \{is\_nat(s^\infty)\} = HB^{co}$
$\text{gfp } f = \{is\_nat(s^n(z)) \mid n \in \mathbb{N}\} \cup \{is\_nat(s^\infty)\} = HB^{co}$
Remark
let s^∞ denote the solution of $X = s(X)$
$is\_nat(s^\infty) \in f(HB^{co})$ since $s(s^\infty) = s^\infty$, $is\_nat(s^\infty) \in HB^{co}$

Inductive and coinductive interpretation with fixed points (3)

Another example

Let f be the one step inference of the following Horn clauses:

$$p(s(N)) \text{ :- } p(N).$$
$$q \text{ :- } p(N).$$

$f^n(\emptyset) = \emptyset$ for all $n \in \mathbb{N}$, hence lfp $f = \emptyset$

$f^1(HB) = \{p(s^k(z)) \mid k \geq 1\} \cup \{q\}$
$f^n(HB) = \{p(s^k(z)) \mid k \geq n\} \cup \{q\}$
$\inf\{f^n(HB) \mid n \in \mathbb{N}\} = \{q\}$, but $f(\{q\}) = \emptyset$, and gfp $(f : \wp(HB) \to \wp(HB)) = \emptyset$

$f^1(HB^{co}) = \{p(s^k(z)) \mid k \geq 1\} \cup \{q, p(s^\infty)\}$
$f^n(HB^{co}) = \{p(s^k(z)) \mid k \geq n\} \cup \{q, p(s^\infty)\}$
$\inf\{f^n(HB) \mid n \in \mathbb{N}\} = \{q, p(s^\infty)\}$
gfp $(f : \wp(HB^{co}) \to \wp(HB^{co})) = \{q, p(s^\infty)\}$

A naive procedure for checking if a ground atom holds

Directly inspired by the Kleene theorem

Inductive interpretation: $p(\bar{t}) \in \text{lfp } f$?

1. $A = \emptyset$
2. if $p(\bar{t}) \in A$ then return `yes`
3. if $f(A) = A$ then return `no`
4. $A = f(A)$
5. repeat from point 2

Coinductive interpretation: $p(\bar{t}) \in \text{gfp } f$?

1. $A = HB^{co}$
2. if $p(\bar{t}) \notin A$ then return `no`
3. if $f(A) = A$ then return `yes`
4. $A = f(A)$
5. repeat from point 2

Problems with this procedure

- it may not terminate
- it computes much more atoms than what is actually required
- the computed sets of atoms are often infinite: a symbolic representation is needed

Proof tree

Intuition: build the least set of ground atoms needed to show that a ground atom holds

Remark: the depth of the proof tree may be infinite for the coinductive interpretation

Example

Functors: $s/1$, $z/0$
Predicate symbols: $geq/2$, $eq/2$
Meta-rules:

$$R1 \quad \frac{}{geq(N,z)}$$

$$R2 \quad \frac{geq(M,N)}{geq(s(M),s(N))}$$

$$R3 \quad \frac{geq(M,N), geq(N,M)}{eq(M,N)}$$

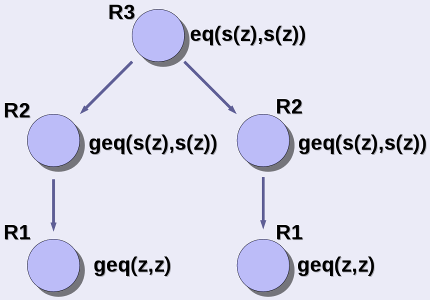
Proof tree

Intuition: build the least set of ground atoms needed to show that a ground atom holds

Remark: the depth of the proof tree may be infinite for the coinductive interpretation

Example

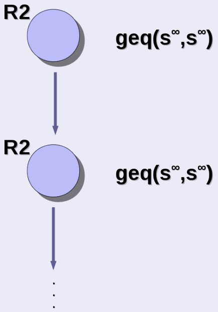
Proof tree showing that $eq(s(z), s(z))$ holds



Infinite proof trees

Example

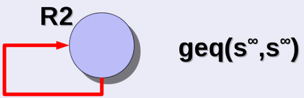
Proof tree showing that $geq(s^\infty, s^\infty)$ holds



Infinite proof trees

Example

In fact, such a proof tree is regular



Proof tree: generalization

Definition of proof tree

- nodes are labeled by ground atoms
- for all nodes n_0, with children $n_1, \ldots, n_k$
  if n_i is labeled by $p_i(\bar{t}_i)$ for all $i = 0, \ldots, k$

  then $\dfrac{p_1(\bar{t}_1), \ldots, p_k(\bar{t}_k)}{p_0(\bar{t}_0)}$ must be an instantiation of a meta-rule

  Remark: if n_0 is a leaf (no children), then such a meta-rule must necessarily be an axiom

Inductive and coinductive interpretation (proof trees)

- inductive interpretation: $\{p(\bar{t}) \in HB \mid$ there is a **finite** proof tree for $p(\bar{t})\}$
- coinductive interpretation: $\{p(\bar{t}) \in HB^{co} \mid$ there is a proof tree for $p(\bar{t})\}$

Example with meta-rules for *is_nat*

$$\frac{}{is\_nat(z)}$$

$$\frac{is\_nat(N)}{is\_nat(s(N))}$$

Inductive interpretation: $\{is\_nat(s^n(z)) \mid n \in \mathbb{N}\}$
Coinductive interpretation: $\{is\_nat(s^n(z)) \mid n \in \mathbb{N}\} \cup \{is\_nat(s^\infty)\}$

Equivalence between fixed point and proof tree interpretation

- $p(\bar{t}) \in \text{lfp}\ (f)$ iff there is a finite proof tree for $p(\bar{t})$
- $p(\bar{t}) \in \text{gfp}\ (f)$ iff there is a proof tree for $p(\bar{t})$

Proof: see [LeroyGrall2009]

Induction and coinduction principle (1)

General claims

- Induction principle: if $f:\wp(U) \to \wp(U)$, f monotone, and S f-closed ($f(S) \subseteq S$), then $\mathrm{lfp}\ f \subseteq S$
- Coinduction principle: if $f:\wp(U) \to \wp(U)$, f monotone, and S f-dense ($S \subseteq f(S)$), then $S \subseteq \mathrm{gfp}\ f$
- Both principles are direct consequences of the Tarski-Knaster theorem
- Proof by induction: if $X = \mathrm{lfp}\ f$, f monotone, then to prove the claim

  $$\forall x \in U, x \in X \Rightarrow x \in S$$

  it is sufficient (but not necessary) to prove that S is f-closed
- Proof by coinduction: if $X = \mathrm{gfp}\ f$, f monotone, then to prove the claim

  $$\forall x \in U, x \in S \Rightarrow x \in X$$

  it is sufficient (but not necessary) to prove that S is f-dense

Induction and coinduction principle (2)

More specific claims for inference systems

- Induction principle
  - f one step inference
  - S f-closed = for all $\frac{p_1(\bar{t}_1), \ldots, p_n(\bar{t}_n)}{p(\bar{t})}$ rules of the system, if $p_1(\bar{t}_1), \ldots, p_n(\bar{t}_n) \in S$, then $p(\bar{t}) \in S$
- Coinduction principle
  - f one step inference
  - S f-dense = for all $p(\bar{t}) \in S$, there exists a rule $\frac{p_1(\bar{t}_1), \ldots, p_n(\bar{t}_n)}{p(\bar{t})}$ of the system, s.t. $p_1(\bar{t}_1), \ldots, p_n(\bar{t}_n) \in S$

Induction principle

Example

Functors: $s/1$, $z/0$
Predicate symbols: $p/1$
Meta-rules:

$$\frac{}{p(z)} \qquad \frac{p(N)}{p(s(s(s(s(N)))))}$$

Let $I = \text{lfp}\ (f{:}\wp(HB) \to \wp(HB))$ (inductive interpretation)

1. $I \subseteq \{p(s^{2n}(z)) \mid n \in \mathbb{N}\}$
2. $I \subseteq \{p(s^{4n}(z)) \mid n \in \mathbb{N}\}$

Both 1 and 2 can be proved by applying the induction principle
Remarks:

- $I \subseteq \{p(s^{2n}(z)) \mid n \in \mathbb{N}\} \cup \{s(z)\}$ and $\{p(s^{4n}(z)) \mid n \in \mathbb{N}\} \subseteq I$ hold, but cannot be directly proved by the induction principle
- $\{p(s^{2n}(z)) \mid n \in \mathbb{N}\} \subseteq I$ **does not hold**

Coinduction principle (1)

Example

Functors: $s/1$, $z/0$
Predicate symbols: $q/1$
Meta-rules:

$$\frac{}{q(z)} \qquad \frac{q(N)}{q(s(s(N)))}$$

Let $I = \text{gfp}\ (f{:}\wp(HB^{co}) \to \wp(HB^{co}))$ (coinductive interpretation)

1. $\{q(s^\infty), q(z)\} \subseteq I$
2. $\{q(s^\infty)\} \cup \{q(s^{2n}(z)) \mid n \in \mathbb{N}\} \subseteq I$

Both 1 and 2 can be proved by applying the coinduction principle
Remarks:

- $\{q(s(s(z)))\} \subseteq I$ and $I \subseteq \{q(s^\infty)\} \cup \{q(s^{2n}(z)) \mid n \in \mathbb{N}\}$ hold, but cannot be directly proved by the coinduction principle
- $I \subseteq \{q(s^\infty), q(z)\}$ **does not hold**

Coinduction principle (2)

Example

Functors: $s/1$, $z/0$
Predicate symbols: $p/1$
Meta-rules:

$$\frac{}{p(z)} \qquad \frac{p(N)}{p(s(s(s(s(N)))))}$$

- Let $I = \text{lfp}\ (f{:}\wp(HB) \to \wp(HB))$ (inductive interpretation)
- $\{p(s^{4n}(z)) \mid n \in \mathbb{N}\} \subseteq I$ can be proved by using the coinduction principle
  1. $\text{lfp}\ (f{:}\wp(HB) \to \wp(HB)) = \text{gfp}\ (f{:}\wp(HB) \to \wp(HB))$, because there exist only finite proof trees
  2. $\{p(s^{4n}(z)) \mid n \in \mathbb{N}\} \subseteq I$ can be proved by the coinduction principle!