Logic & Constraint Prog.

Syntax and semantics of Prolog programms

IENAC S

Sept. 2015

Basic syntactical constructs

atom: a name used to name relations and data objects; it can be

- a sequence of letters, digits that starts with a lowercase letter, and can also contain the underscore "_"
- a sequence of characters enclosed between two single quotes

Examples: laysan_albatross dark21 'The Blues Brothers'

variable: string of letters, digits, "_" that starts with an uppercase letter, or with "_"

Examples: X1 Toto12_urt___cur4 _123urc_

numerical constants: usual representations for (signed) integers and floating point numbers

term: an expression that represents a *data object*:
atoms, numerical constants, variables
+ compound terms of the form $f(t_1, \ldots, t_n)$
where f is an atom and t_1, ..., t_n are terms
(Lists are a special kind of compound terms.)

goal: expression that can be *true* of *false*;
has the form $p(t_1, \ldots, t_n)$
where p is an atom and t_1, ..., t_n are terms.

Logical constructs

formula: *goals* assembled with connectives
$\wedge$ (conjunction), $\vee$ (disjunction), $\neg$ (negation).

rule: $$\underbrace{G}_{\text{head}} \leftarrow \underbrace{\varphi}_{\text{body}}.$$
where G is a goal and φ is a formula.

fact: rule with $\varphi = \top$ (always true); writtten G.

clause: rule of the form $G \leftarrow L_1 \wedge \ldots \wedge L_m$
where $L_1, \ldots L_m$ are *literals*,
that is, goals and negated goals. (Thus a Prolog clause does not contain any disjunction.)

Logical constructs : Quantification

$\mathsf{directed}(D, A) \leftarrow \mathsf{director}(D, M) \wedge \mathsf{cast}(M, A, R).$ is read:
"*for all* D, A, D has directed A if *there exists* some M, the director of which is D, and in which A played"
In logic, we would write:
$\forall D, A(\mathsf{directed}(D, A) \leftarrow \exists M(\mathsf{director}(D, M) \wedge \mathsf{cast}(M, A, R)))$

- the variables that appear in the head of a rule have an implicit *universal* quantification / meaning
  it is understood that the rule is true for all possible values of these variables
- the variables that appear <u>only</u> in the body of a rule have an implicit *existential* quantification/meaning within the body of the rule
  it is understood that the head of the rule is true if there is at least one value for each of these variables for which the body of the rule is true

Logical constructs : Negation as failure

Procedural meaning given to negation:

In order to prove $\neg\varphi$, try to prove that φ *cannot be proved*.

Consider the movie database excerpt above, and a predicate defining movies in which played actors who were never directed by John Landis:

$p(M) \leftarrow \mathsf{cast}(M, A, R) \wedge \neg\mathsf{directed}(\text{'\textit{John Landis}'}, A).$

Draw the search tree for the query $p(\text{'\textit{Soul Kitchen}'})?$.

Logical constructs : Negation as failure

Procedural meaning given to negation:

In order to prove $\neg\varphi$, try to prove that φ *cannot be proved*.

Consider the movie database excerpt above, and a predicate defining movies in which played actors who were never directed by John Landis:

$p(M) \leftarrow \mathsf{cast}(M, A, R) \land \neg\mathsf{directed}(\textit{'John Landis'}, A).$

Draw the search tree for the query $p(\textit{'Soul Kitchen'})?$.

Negation as failure is <u>not</u> logical negation !

The above definition is *logically* equivalent to:

$p(M) \leftarrow \neg\mathsf{directed}(\textit{'John Landis'}, A) \land \mathsf{cast}(M, A, R).$

But if we submit the query $p(\textit{'Soul Kitchen'})?\ldots$

Predicates and programs

predicate (or relation): characterized by it *name* and its *arity* (number of arguments);
the predicate of name p and arity n, denoted p/n, is defined by a set of *rules* / *facts* of the form:
$p(t_1, \ldots, t_n) \leftarrow \varphi$ or $p(t_1, \ldots, t_n)$.
Predicates are to Prolog what functions / procedures are to more conventional programming languages.

Predicates and programs

predicate (or relation): characterized by it *name* and its *arity* (number of arguments);
the predicate of name p and arity n, denoted p/n, is defined by a set of *rules / facts* of the form:
$p(t_1, \ldots, t_n) \leftarrow \varphi$ or $p(t_1, \ldots, t_n)$.
Predicates are to Prolog what functions / procedures are to more conventional programming languages.

Remark Any rule is equivalent to as set of clauses
because of properties of $\neg$, $\wedge$, $\vee$ (Boolean algebra), and because:
$\psi \leftarrow \varphi_1 \vee \varphi_2$ is equivalent to $\left\{ \begin{array}{l} \psi \leftarrow \varphi_1 \\ \psi \leftarrow \varphi_2 \end{array} \right\}$
and
$\psi \leftarrow \neg\varphi$ is equivalent to $\left\{ \begin{array}{l} \psi \leftarrow \neg q(Y) \\ q(Y) \leftarrow \varphi \end{array} \right\}$
where q is a “new” predicate, and Y is the (sequence of) variable(s) that appear in φ and not in ψ

Predicates and programs

logic program: a set of definitions of predicates.

> **Remark** Clauses or rules that define a predicate p/n must not be interleaved with rules or clauses that define other predicates. (If the definition of predicate p is scattered at different places in a file, Prolog considers that they are successive definitions of the predicate p, each definition canceling the previous one.)

query: a formula.

Given a logic program $\mathbf{P}$ and a query φ, let U be the vector of all the variables that appear in φ and that are not only within the scope of a negation:
we want to know what are the values of U for which $\mathbf{P} \vdash \varphi$.
(But $\vdash$ is not exactly logical deduction.)