Definitions - ILP

Simon Jacquet

August 30, 2022

Contents

1 Predicate logic

1.1 Fundamentals

Variable: Xxxx

Function: xxxx(<Term>,...,<Term>)
Operator that maps inputs to some output. Terms as input, terms as output $(2 + 3)$.

Predicate: xxxx(<Term>,...,<Term>)
Function that takes terms as input and outputs either true or false $(2 < 3)$.

Function symbol or predicate symbol: xxxx

Constant: xxxx
Function symbol or predicate symbol with arity 0.

Term: <Constant> | <Variable> | <Function symbol>

Atom (or atomic formula): Predicate symbol (ntuple of terms).

Literal: <Atom> | ¬ <Atom>
A literal is either an atom or its negation. Let A be an atom,

- A is a positive literal;
- $\neg A$ (or $\overline{A}$) is a negative literal.

1.2 Clauses

Clause: (<Literal>,...,<Literal>)
A clause is a finite set of literals. It is to be taken as the disjunction of the literals. Let A_i, B_i be atoms, the same clause can be written as:

$$\begin{aligned} & (A_1, ..., A_n, \neg B_1, ..., \neg B_m) \\ \iff \quad & A_1 \vee ... \vee A_n \vee B_1 \vee ... \vee B_m \\ \iff \quad & A_1, ..., A_n \leftarrow B_1, ..., B_m \end{aligned}$$

Horn clause: $A \leftarrow B_1, ..., B_m \quad | \quad \leftarrow B_1, ..., B_m$
A Horn clause has at most 1 positive literal. A Horn clause is either a goal or a definite clause.

Denial or goal: $\leftarrow B_1, ..., B_m$
A denial or goal is a Horn clause with 0 positive literal.

Definite clause: $A \leftarrow B_1, ..., B_m$
A Horn clause with exactly 1 positive literal. A, the positive literal is called the head. $B_1, ..., B_n$, the negative literals are called the body.

Unit clause: $A \leftarrow \quad | \quad \leftarrow B$
A unit clause is a Horn clause composed of a single literal, either a positive literal ($A \leftarrow$) or a negative literal ($\leftarrow B$).

1.3 Clausal theory

Clausal theory (aka logic program): (<Clause>,...,<Clause>)
A clausal theory is a set, or conjunction of clauses. Let C_i be clauses, a clausal theory can be written as:

$$\begin{aligned} & (C_1, ..., C_n) \\ \iff \quad & C_1 \wedge ... \wedge C_n \end{aligned}$$

Monoadic (Dyadic) clausal theory: Clausal theory in which all predicates have arity ≤ 1 (≤ 2).

Horn logic program: Clausal theory in which all clauses are Horn clauses.

Definite logic program: Clausal theory in which all clauses are definite clauses.

Datalog program: Logic program in which there are no functions, with the exception of constants.

Higher-order Datalog program: Datalog program in which there is a predicate which has a predicate as argument.

Well-formed-formulaes (wffs): <Literal> | <Clause> | <Clausal theory>

Ground: Let E be a wff or a term, E is ground iff it contains no variable.

Skolemisation: Replacing variables by constants.

Skolem constants: Unique constants.

2 Grammar

2.1 Fundamentals

Σ	Finite alphabet
Σ^*	Infinite set of strings containing ≥ 0 letters from Σ
λ	Empty string
uv	Concatenation of string u and v
$\lvert u \rvert$	Length of string u
L	Language, a subset of $\Sigma*$
ν	Set of non-terminal symbols disjoint from Σ
r	Production rule $LHS \to RHS$ Well-formed when $LHS \in (\nu \cup \Sigma)^*$, $RHS \in (\nu \cup \Sigma \cup \lambda)^*$ When applied, replaces LHS by RHS in a given string
G	Grammar composed of the pair $< s, R >$
s	Start symbol, $s \in \nu$
R	Finite set of production rules
$L(G)$	Language that follows grammar G

2.2 Types of grammar

Let $a, b \in \Sigma$, let $S, A, B, C \in \nu$.

Regular Chomsky-normal grammar: $S \to \lambda \quad | \quad S \to aB$
Production rules must be of the form above.

Linear Context-Free grammar: $S \to \lambda \quad | \quad S \to aB \quad | \quad S \to Ab$
Production rules must be of the form above.

Context-Free grammar: $S \to \lambda \quad | \quad S \to aB \quad | \quad S \to Ab \quad | \quad S \to AB$
Production rules must be of the form above.

Deterministic Context-Free grammar: Context-Free grammar with no two rules $S \to aB, \quad S \to aC$ with $B \neq C$.

2.3 Properties of language

- $\sigma \in \Sigma^*$ is in $L(G) \iff \begin{cases} \exists s \in \nu, \\ \exists (r_1, ..., r_n) \in R^n, \\ s \to_{r_1} ... \to_{r_n} \sigma. \end{cases}$
- Language L is Regular/Linear Context-Free/Context-Free iff $\exists G$, $L = L(G)$, with G, a Regular/Linear Context-Free/Context-Free grammar.