Outline

Inference Systems

Inference System

- inference has the form

$$\frac{F_1 \quad \ldots \quad F_n}{G},$$

where $n \geq 0$ and $F_1, \ldots, F_n, G$ are formulas.
- The formula G is called the conclusion of the inference;
- The formulas $F_1, \ldots, F_n$ are called its premises.
- An inference rule R is a set of inferences.
- Every inference $I \in R$ is called an instance of R.
- An Inference system $\mathbb{I}$ is a set of inference rules.
- Axiom: inference rule with no premises.

Inference System: Example

Represent the natural number n by the string $\underbrace{|\ldots|}_{n \text{ times}}\varepsilon$.

The following inference system contains 6 inference rules for deriving equalities between expressions containing natural numbers, addition $+$ and multiplication $\cdot$.

$$\frac{}{\varepsilon = \varepsilon}\ (\varepsilon) \qquad \frac{x = y}{|x = |y}\ (|)$$

$$\frac{}{\varepsilon + x = x}\ (+_1) \qquad \frac{x + y = z}{|x + y = |z}\ (+_2)$$

$$\frac{}{\varepsilon \cdot x = \varepsilon}\ (\cdot_1) \qquad \frac{x \cdot y = u \quad y + u = z}{|x \cdot y = z}\ (\cdot_2)$$

Derivation, Proof

- Derivation in an inference system $\mathbb{I}$: a tree built from inferences in $\mathbb{I}$.
- If the root of this derivation is E, then we say it is a derivation of E.
- Proof of E: a finite derivation whose leaves are axioms.
- Derivation of E from $E_1, \ldots, E_m$: a finite derivation of E whose every leaf is either an axiom or one of the expressions $E_1, \ldots, E_m$.

Examples

For example,

$$\frac{||\varepsilon + |\varepsilon = |||\varepsilon}{|||\varepsilon + |\varepsilon = ||||\varepsilon}\ (+_2)$$

is an inference that is an instance (special case) of the inference rule

$$\frac{x + y = z}{|x + y = |z}\ (+_2)$$

Examples

For example,

$$\frac{||\varepsilon + |\varepsilon = |||\varepsilon}{|||\varepsilon + |\varepsilon = ||||\varepsilon}\ (+_2)$$

is an inference that is an instance (special case) of the inference rule

$$\frac{x + y = z}{|x + y = |z}\ (+_2)$$

It has one premise $||\varepsilon + |\varepsilon = |||\varepsilon$ and the conclusion $|||\varepsilon + |\varepsilon = ||||\varepsilon$.

Examples

For example,

$$\frac{||\varepsilon + |\varepsilon = |||\varepsilon}{|||\varepsilon + |\varepsilon = ||||\varepsilon}\ (+_2)$$

is an inference that is an instance (special case) of the inference rule

$$\frac{x + y = z}{|x + y = |z}\ (+_2)$$

It has one premise $||\varepsilon + |\varepsilon = |||\varepsilon$ and the conclusion $|||\varepsilon + |\varepsilon = ||||\varepsilon$.

The axiom

$$\frac{}{\varepsilon + |||\varepsilon = |||\varepsilon}\ (+_1)$$

is an instance of the rule

$$\frac{}{\varepsilon + x = x}\ (+_1)$$

Proof in this Inference System

Proof of $||\varepsilon \cdot ||\varepsilon = ||||\varepsilon$ (that is, $2 \cdot 2 = 4$).

$$
\dfrac{
  \dfrac{
    \dfrac{}{\varepsilon \cdot ||\varepsilon = \varepsilon}\,(\cdot_1)
    \qquad
    \dfrac{
      \dfrac{
        \dfrac{}{\varepsilon + \varepsilon = \varepsilon}\,(+_1)
      }{|\varepsilon + \varepsilon = |\varepsilon}\,(+_2)
    }{||\varepsilon + \varepsilon = ||\varepsilon}\,(+_2)
  }{|\varepsilon \cdot ||\varepsilon = ||\varepsilon}\,(\cdot_2)
  \qquad
  \dfrac{
    \dfrac{
      \dfrac{}{\varepsilon + ||\varepsilon = ||\varepsilon}\,(+_1)
    }{|\varepsilon + ||\varepsilon = |||\varepsilon}\,(+_2)
  }{||\varepsilon + ||\varepsilon = ||||\varepsilon}\,(+_2)
}{||\varepsilon \cdot ||\varepsilon = ||||\varepsilon}\,(\cdot_2).
$$

Derivation in this Inference System

Derivation of $||\varepsilon \cdot ||\varepsilon = |||||\varepsilon$ from $\varepsilon + ||\varepsilon = |||\varepsilon$ (that is, $2 + 2 = 5$ from $0 + 2 = 3$).

$$\dfrac{\dfrac{\dfrac{}{\varepsilon \cdot ||\varepsilon = \varepsilon}\;(\cdot_1) \qquad \dfrac{\dfrac{\dfrac{}{\varepsilon + \varepsilon = \varepsilon}\;(+_1)}{|\varepsilon + \varepsilon = |\varepsilon}\;(+_2)}{||\varepsilon + \varepsilon = ||\varepsilon}\;(+_2)}{|\varepsilon \cdot ||\varepsilon = ||\varepsilon}\;(\cdot_2) \qquad \dfrac{\dfrac{\varepsilon + ||\varepsilon = |||\varepsilon}{|\varepsilon + ||\varepsilon = ||||\varepsilon}\;(+_2)}{||\varepsilon + ||\varepsilon = |||||\varepsilon}\;(+_2)}{||\varepsilon \cdot ||\varepsilon = ||||\varepsilon}\;(\cdot_2).$$

Arbitrary First-Order Formulas

- A first-order signature (vocabulary): function symbols (including constants), predicate symbols. Equality is part of the language.
- A set of variables.
- Terms are buit using variables and function symbols. For example, $f(x) + g(x)$.
- Atoms, or atomic formulas are obtained by applying a predicate symbol to a sequence of terms. For example, $p(a, x)$ or $f(x) + g(x) \geq 2$.
- Formulas: built from atoms using logical connectives $\neg$, $\wedge$, $\vee$, $\rightarrow$, $\leftrightarrow$ and quantifiers $\forall$, $\exists$. For example, $(\forall x)x = 0 \vee (\exists y)y > x$.

Clauses

- Literal: either an atom A or its negation $\neg A$.
- Clause: a disjunction $L_1 \vee \ldots \vee L_n$ of literals, where $n \geq 0$.

Clauses

- Literal: either an atom A or its negation $\neg A$.
- Clause: a disjunction $L_1 \vee \ldots \vee L_n$ of literals, where $n \geq 0$.
- Empty clause, denoted by $\square$: clause with 0 literals, that is, when $n = 0$.

Clauses

- Literal: either an atom A or its negation $\neg A$.
- Clause: a disjunction $L_1 \vee \ldots \vee L_n$ of literals, where $n \geq 0$.
- Empty clause, denoted by $\square$: clause with 0 literals, that is, when $n = 0$.
- A formula in Clausal Normal Form (CNF): a conjunction of clauses.

Clauses

- Literal: either an atom A or its negation $\neg A$.
- Clause: a disjunction $L_1 \vee \ldots \vee L_n$ of literals, where $n \geq 0$.
- Empty clause, denoted by $\square$: clause with 0 literals, that is, when $n = 0$.
- A formula in Clausal Normal Form (CNF): a conjunction of clauses.
- A clause is ground if it contains no variables.
- If a clause contains variables, we assume that it implicitly universally quantified. That is, we treat $p(x) \vee q(x)$ as $\forall x(p(x) \vee q(x))$.

Binary Resolution Inference System

The binary resolution inference system, denoted by $\mathbb{BR}$ is an inference system on propositional clauses (or ground clauses). It consists of two inference rules:

- Binary resolution, denoted by BR:

$$\frac{p \vee C_1 \quad \neg p \vee C_2}{C_1 \vee C_2} \text{ (BR)}.$$

- Factoring, denoted by Fact:

$$\frac{L \vee L \vee C}{L \vee C} \text{ (Fact)}.$$

Soundness

- An inference is sound if the conclusion of this inference is a logical consequence of its premises.
- An inference system is sound if every inference rule in this system is sound.

Soundness

- An inference is sound if the conclusion of this inference is a logical consequence of its premises.
- An inference system is sound if every inference rule in this system is sound.

$\mathbb{BR}$ is sound.

Consequence of soundness: let S be a set of clauses. If $\square$ can be derived from S in $\mathbb{BR}$, then S is unsatisfiable.

Example

Consider the following set of clauses

$$\{\neg p \vee \neg q,\ \neg p \vee q,\ p \vee \neg q,\ p \vee q\}.$$

The following derivation derives the empty clause from this set:

$$\dfrac{\dfrac{\dfrac{p \vee q \quad p \vee \neg q}{p \vee p}\ \text{(BR)}}{p}\ \text{(Fact)} \qquad \dfrac{\dfrac{\neg p \vee q \quad \neg p \vee \neg q}{\neg p \vee \neg p}\ \text{(BR)}}{\neg p}\ \text{(Fact)}}{\square}\ \text{(BR)}$$

Hence, this set of clauses is unsatisfiable.

Can this be used for checking (un)satisfiability

1. What happens when the empty clause cannot be derived from S?
2. How can one search for possible derivations of the empty clause?

Can this be used for checking (un)satisfiability

1. Completeness.

   Let S be an unsatisfiable set of clauses. Then there exists a derivation of $\Box$ from S in $\mathbb{BR}$.

Can this be used for checking (un)satisfiability

1. Completeness.

   Let S be an unsatisfiable set of clauses. Then there exists a derivation of $\square$ from S in $\mathbb{BR}$.

2. We have to formalize search for derivations.

However, before doing this we will introduce a slightly more refined inference system.

Selection Function

A literal selection function selects literals in a clause.

- If C is non-empty, then at least one literal is selected in C.

Selection Function

A literal selection function selects literals in a clause.

- If C is non-empty, then at least one literal is selected in C.

We denote selected literals by underlining them, e.g.,

$$\underline{p} \vee \neg q$$

Selection Function

A literal selection function selects literals in a clause.

- If C is non-empty, then at least one literal is selected in C.

We denote selected literals by underlining them, e.g.,

$$\underline{p} \vee \neg q$$

Note: selection function does not have to be a function. It can be any oracle that selects literals.

Binary Resolution with Selection

We introduce a family of inference systems, parametrised by a literal selection function σ.
The binary resolution inference system, denoted by $\mathbb{BR}_\sigma$, consists of two inference rules:

- Binary resolution, denoted by BR

$$\frac{\underline{p} \vee C_1 \quad \underline{\neg p} \vee C_2}{C_1 \vee C_2} \text{ (BR).}$$

Binary Resolution with Selection

We introduce a family of inference systems, parametrised by a literal selection function σ.
The binary resolution inference system, denoted by $\mathbb{BR}_\sigma$, consists of two inference rules:

- Binary resolution, denoted by BR

$$\frac{\underline{p} \vee C_1 \quad \underline{\neg p} \vee C_2}{C_1 \vee C_2} \ (\mathsf{BR}).$$

- Positive factoring, denoted by Fact:

$$\frac{\underline{p} \vee \underline{p} \vee C}{p \vee C} \ (\mathsf{Fact}).$$

Completeness?

Binary resolution with selection may be incomplete, even when factoring is unrestricted (also applied to negative literals).

Completeness?

Binary resolution with selection may be incomplete, even when factoring is unrestricted (also applied to negative literals).

Consider this set of clauses:

$$
\begin{array}{ll}
(1) & \neg q \vee \underline{r} \\
(2) & \neg p \vee \underline{q} \\
(3) & \neg r \vee \underline{\neg q} \\
(4) & \neg q \vee \underline{\neg p} \\
(5) & \neg p \vee \underline{\neg r} \\
(6) & \neg r \vee \underline{p} \\
(7) & r \vee q \vee \underline{p}
\end{array}
$$

Completeness?

Binary resolution with selection may be incomplete, even when factoring is unrestricted (also applied to negative literals).

Consider this set of clauses:

$$
\begin{array}{ll}
(1) & \neg q \lor \underline{r} \\
(2) & \neg p \lor \underline{q} \\
(3) & \neg r \lor \underline{\neg q} \\
(4) & \neg q \lor \underline{\neg p} \\
(5) & \neg p \lor \underline{\neg r} \\
(6) & \neg r \lor \underline{p} \\
(7) & r \lor q \lor \underline{p}
\end{array}
$$

It is unsatisfiable:

$$
\begin{array}{lll}
(8) & q \lor p & (6,7) \\
(9) & q & (2,8) \\
(10) & r & (1,9) \\
(11) & \neg q & (3,10) \\
(12) & \square & (9,11)
\end{array}
$$

Note the linear representation of derivations (used by Vampire and many other provers).

However, any inference with selection applied to this set of clauses give either a clause in this set, or a clause containing a clause in this set.

Literal Orderings

Take any well-founded ordering $\succ$ on atoms, that is, an ordering such that there is no infinite decreasing chain of atoms:

$$A_0 \succ A_1 \succ A_2 \succ \cdots$$

In the sequel $\succ$ will always denote a well-founded ordering.

Literal Orderings

Take any well-founded ordering $\succ$ on atoms, that is, an ordering such that there is no infinite decreasing chain of atoms:

$$A_0 \succ A_1 \succ A_2 \succ \cdots$$

In the sequel $\succ$ will always denote a well-founded ordering.

Extend it to an ordering on literals by:

- If $p \succ q$, then $p \succ \neg q$ and $\neg p \succ q$;
- $\neg p \succ p$.

Literal Orderings

Take any well-founded ordering $\succ$ on atoms, that is, an ordering such that there is no infinite decreasing chain of atoms:

$$A_0 \succ A_1 \succ A_2 \succ \cdots$$

In the sequel $\succ$ will always denote a well-founded ordering.

Extend it to an ordering on literals by:

- If $p \succ q$, then $p \succ \neg q$ and $\neg p \succ q$;
- $\neg p \succ p$.

Exercise: prove that the induced ordering on literals is well-founded too.

Orderings and Well-Behaved Selections

Fix an ordering $\succ$. A literal selection function is well-behaved if

- If all selected literals are positive, then all maximal (w.r.t. $\succ$) literals in C are selected.

In other words, either a negative literal is selected, or all maximal literals must be selected.

Orderings and Well-Behaved Selections

Fix an ordering $\succ$. A literal selection function is well-behaved if

- If all selected literals are positive, then all maximal (w.r.t. $\succ$) literals in C are selected.

In other words, either a negative literal is selected, or all maximal literals must be selected.

To be well-behaved, we sometimes must select more than one different literal in a clause. Example: $p \vee p$ or $p(x) \vee p(y)$.

Completeness of Binary Resolution with Selection

Binary resolution with selection is complete for every well-behaved selection function.

Completeness of Binary Resolution with Selection

Binary resolution with selection is complete for every well-behaved selection function.

Consider our previous example:

$$
\begin{array}{ll}
(1) & \neg q \vee \underline{r} \\
(2) & \neg p \vee \underline{q} \\
(3) & \neg r \vee \underline{\neg q} \\
(4) & \neg q \vee \underline{\neg p} \\
(5) & \neg p \vee \underline{\neg r} \\
(6) & \neg r \vee \underline{p} \\
(7) & r \vee q \vee \underline{p}
\end{array}
$$

A well-behave selection function must satisfy:

1. $r \succ q$, because of (1)
2. $q \succ p$, because of (2)
3. $p \succ r$, because of (6)

There is no ordering that satisfies these conditions.