Semantics for First-Order Logic:

The correctness of a logical derivation $\Gamma \vdash \Delta$ is independent of the meanings of the non-logical symbols. (Non-logical symbols are the predicate and functional symbols)

The fact that a logical derivation $\Gamma \vdash \Delta$ is correct means: Whenever the formulas in Γ are true, one of the formulas in Δ is true.

When non-logical symbols are interpreted arbitrarily, in most cases, not all formulas in Γ are true.

In practice, one is more interested in situations where the non-logical symbols have a meaningful interpretation and all formulas in Γ are true. In that case one can learn something from a proof of $\Gamma \vdash \Delta$.

Semantics for First-Order Logic

An interpretation of $\mathcal{P}$, $\mathcal{F}$ is an ordered pair $(D, [\ ])$, s.t.

- D is a non-empty set. The domain or universe.
- For each $p \in \mathcal{P}$ with arity n, $[p]$ is a subset of D^n.
- For each $f \in \mathcal{F}$ with arity n, $[f]$ is a function from D^n to D.

- Whenever a logician says 'function', he means 'total function'. This is not because there is something wrong with partial functions, but because there is no elegant way of handling partial functions in logic.
- The non-emptiness condition of D is ugly. It is also unnatural in the case of many-typed logic. (Because the fact that somebody invents some type does not imply that there is something in it) It is, however, standard and we stick with the standard.
- For predicates p with arity 0, we assume that D^0 consists of a single tuple ( ). This tuple either is in $[p]$, or is not in $[p]$.

Truth of Atoms in an Interpretation

In an interpretation $(D, [\,])$, each term over $\mathcal{F}$ can be mapped to an element of D as follows:

- For a term $f(t_1, \ldots, t_n)$ with $n > 0$, $[f(t_1, \ldots, t_n)] = [f]([t_1], \ldots, [t_n])$.

The truth of an atom in an interpretation $(D, [\,])$ is defined as follows:

- An atom $p(t_1, \ldots, t_n)$ is true in $(D, [\,])$ iff $([t_1], \ldots, [t_n]) \in [p]$.
- An atom $t_1 \approx t_2$ is true in $(D, [\,])$ iff $[t_1] = [t_2]$.

Truth of a Formula in an Interpretation

We recursively define when a formula F is true in an interpretation $(D, [\,])$. (Written as $(D, [\,]) \models F$)

- $\perp$ is never true in $(D, [\,])$.
- $\top$ is always true in $(D, [\,])$.
- $\neg F$ is true in $(D, [\,])$ iff F is false in $(D, [\,])$.
- $F_1 \vee F_2$ is true in $(D, [\,])$ iff F_1 is true in $(D, [\,])$ or F_2 is true in $(D, [\,])$.
- $F_1 \wedge F_2$ is true in $(D, [\,])$ iff F_1 is true in $(D, [\,])$ and F_2 is true in $(D, [\,])$.

Truth of a Formula in an Interpretation

- $F_1 \to F_2$ is true in $(D, [\ ])$ iff F_1 is false in $(D, [\ ])$ or F_2 is true in $(D, [\ ])$.
- $F_1 \leftrightarrow F_2$ is true in $(D, [\ ]$ iff either F_1, F_2 are both true in $(D, [\ ])$ or F_1, F_2 are both false in $(D, [\ ])$.
- $\forall x\ F$ is true in $(D, [\ ])$ iff for all $d \in D$, the formula F is true in the modified interpretation $(D, [\ ]^x_d)$.
- $\exists x\ F$ is true in $(D, [\ ])$ iff there is a $d \in D$, for which the formula F is true in the modified interpretation $(D, [\ ]^x_d)$.

The modified assignment $[\ ]_d^x$ (which was used for definition truth of $\forall$ and $\exists$) is defined as follows:

- For all $p \in \mathcal{P}$, $\quad [p]_d^x = [p]$.
- For all $f \in \mathcal{F}$, s.t $f \neq x$, $\quad [f]_d^x = [f]$.
- For x, $\quad [x]_d^x = d$.

In words: The modified assignment $[\ ]_d^x$ is the same as $[\ ]$ everywhere, except possibly in x, where it takes the value d.

Example

Suppose we have two constants a, b, one function f, and one predicate P.

Let's take $D = \{0, 1\}$. We could take $[a] = 0,\ [b] = 1$, and define $[f](0) = 1,\ \ [f](1) = 0$. Furthermore, we could define $[P] = \{(0, 1),\ (1, 1)\}$.

Then

$$[f(f(a))] = 0, \quad [f(f(f(b)))] = 0, \quad [f(f(b))] = 1.$$

$$(D, [\,]) \not\models \exists x\ P(x, x), \quad (D, [\,]) \models \exists x \exists y\ P(x, y).$$

$$(D, [\,]) \models \forall x \exists y\ P(x, y), \quad (D, [\,]) \not\models \forall x \exists y\ (P(x, y) \wedge x \neq y).$$

Definition: A sequent $\Gamma \vdash \Delta$ is valid, if

for every interpretation $(D, [\ ])$, for which
for all $A \in \Gamma \quad (D, [\ ]) \models A$,
there is a $B \in \Delta$ s.t. $(D, [\ ]) \models B$.

THEOREM:

The sequent $\Gamma \vdash \Delta$ is provable if and only if it is valid.

The fact that (1) implies (2) is called soundness. The fact that (2) implies (1) is called completeness.

In general, soundness is more important than completeness. In many applications, an incomplete calculus is acceptable, while an unsound calculus is useless.

Example:

We suspect that the sequent $\exists x\ P(x) \vdash \forall x\ P(x)$ has no proof. If we can find an interpretation $(D, [\,])$ s.t $(D, [\,]) \models \exists x\ P(x)$, but not $(D, [\,]) \models \forall x\ P(x)$, then we can be certain.

Put $D = \{c_0, c_1\}$, and $[P] = \{c_0\}$.

Because $[x]^x_{c_0} = c_0$ and $c_0 \in [P]$, we have $(D, [\,]^x_{c_0}) \models P(x)$.

Because $[x]^x_{c_1} = c_1$ and $c_1 \notin [P]$, we have $(D, [\,]^x_{c_1}) \not\models P(x)$.

Therefore $(D, [\,]) \models \exists x\ P(x)$, but $(D, [\,]) \not\models \forall x\ P(x)$.

Excercise: Show that there is no counter model with only one element.

Soundness of Sequent Calculus

In order to show soundness of sequent calculus, observe that we

1. want to show validity for every provable sequent,
2. the set of provable sequents is an inductively defined set.

Therefore it is sufficient to show that validity is preserved by each of the rules of sequent calculus.

For the propositional rules, the soundness proof essentially the same as for propositional logic.

Soundness of Axiom

Consider the axiom $\Gamma, A \vdash \Delta, A$. If for each $F \in \Gamma, A, \quad (D, [\,]) \models F$ then clearly there is an $F \in \Delta, A$ s.t. $(D, [\,]) \models F$.

Assume that two sequents $S_1 = \ \Gamma, A \vdash \Delta$ and $S_2 = \ \Gamma \vdash \Delta, A$ are valid. We want to show that $\Gamma \vdash \Delta$ is also valid. This is done by assuming that for every $\gamma \in \Gamma, \quad (D, [\,]) \models \gamma$ and showing that there is a $\delta \in \Delta$ with $(D, [\,]) \models \delta$.

Assume that for each $\gamma \in \Gamma, \quad (D, [\,]) \models \gamma$. Then, by validity of S_2, there is a δ in Δ, A, such that $(D, [\,]) \models \delta$. If this δ happens to be in Δ, then we are done.

Otherwise it must be the case that $\delta = A$. Then by validity of S_1, it must be the case that there is a $\delta' \in \Delta$ for which $(D, [\,]) \models \delta'$.

Soundness of $\wedge$-left

Assume that $\Gamma, A, B \vdash \Delta$ is valid.

We want to show that $\Gamma, A \wedge B \vdash \Delta$ is also valid. Assume that for each $\gamma \in \Gamma, A \wedge B$, we have $(D, [\,]) \models \gamma$. From this follows:

1. For each $\gamma \in \Gamma, \quad (D, [\,]) \models \gamma$,
2. $(D, [\,]) \models A \wedge B$,

From 2 follows that $(D, [\,]) \models A$ and $(D, [\,]) \models B$. Then, using validity of $\Gamma, A, B \vdash \Delta$, we see that there is a $\delta \in \Delta$ for which $(D, [\,]) \models \delta$.

Soundness of $\forall$-right

Assume that the sequent $\Gamma \vdash \Delta, P$ is a valid sequent.

We have to show show that $\Gamma \vdash \Delta,\ \forall x\ P$ is a valid sequent as well.

Let $(D, [\,])$ be an arbitrary interpretation which makes all formulas $A \in \Gamma$ true. We show that $(D, [\,])$ makes either a formula in Δ, or $\forall x\ P$ true.

We assume that $(D, [\,])$ is an interpretation that makes all formulas in Γ true and no formula in Δ. If we can show that $(D, [\,])$ makes $\forall x\ P$ true, then the proof is complete.

Now let I' be an arbitrary interpretation of form $(D, []_d^x)$. Because x is not free in Γ or Δ, and $\Gamma \vdash \Delta,\ P$ is a valid sequent, I', makes P true.

Since every interpretation of form $(D, [\,]_d^x)$ makes P true, it follows that $(D, [\,])$ makes $\forall x\ P$ true.

Soundness of $\forall$-left

We need the following property which states that the thruth of a formula does not depend on the variables that are not free in it.

Theorem 1: Let F be a formula. Let x be a variable which is not free in F. Let $(D, [\,])$ be an interpretation. Let $d_1, d_2 \in D$. Then $(D, [\,]^x_{d_1}) \models F$ iff $(D, [\,]^x_{d_2}) \models F$.

Theorem 1 can be proven by induction on the structure of F. First one proves (by induction on the term structure) a lemma stating that for every t, if x does not occur in t, then $[t]^x_{d_1} = [t]^x_{d_2}$.

Proof of Theorem 1

- Let $p(t_1, \ldots, t_n)$ be an atom not containing x. Then $(D, [\ ]^x_{d_1}) \models p(t_1, \ldots, t_n)$ iff $([t_1]^x_{d_1}, \ldots, [t_n]^x_{d_1}) \in [p]^x_{d_1}$. By the lemma of the previous slide, this is equivalent to $([t_1]^x_{d_2}, \ldots, [t_n]^x_{d_2}) \in [p]^x_{d_2}$, which is the definition of $(D, [\ ]^x_{d_2}) \models p(t_1, \ldots, t_n)$.

- The propositional operators are straightforward.

- Assume that F has form $\forall x\ P$. Then $(D, [\ ]^x_{d_1}) \models \forall x\ P$ iff for all $d \in D$, $(D, [\ ]^x_{d_1}{}^x_d) \models P$. Since the second assignment ${}^x_d$ overwrites the first assignment ${}^x_{d_1}$, this is equivalent to: for all $d \in D$, $(D, [\ ]^x_{d_2}{}^x_d) \models P$, which in turn is equivalent to $(D, [\ ]^x_{d_2}) \models \forall x\ P$.

- Assume that F has form $\forall y\ P$ and $x \neq y$. Then, by definition, $(D, [\ ]^{x}_{d_1}) \models \forall y\ P$ iff for all $d \in D$, $(D, [\ ]^{x\ \ y}_{d_1\ d}) \models P$. Because x and y are distinct, this is equivalent to $(D, [\ ]^{y\ \ x}_{d\ d_1}) \models P$. Using the induction hypothesis, this is equivalent to $(D, [\ ]^{y\ \ x}_{d\ d_2}) \models P$. Since this is equivalent to: For all $d \in D$, $(D, [\ ]^{x\ \ y}_{d_1\ d}) \models P$, this is equivalent to $(D, [\ ]^{y}_{d}) \models \forall y\ P$.
- The cases for $\exists$ are analogeous.

Theorem 2: Let F be a formula. Let $(D, [\ ])$ be an interpretation. Let x be a variable. Let t be a term.

Then $(D, [\ ]^x_{[t]}) \models F$ iff $(D, [\ ]) \models F[x := t]$.

Theorem 2 is proven by induction on the formula structure of F. First we show the following property: For every term u, $[u]^x_{[t]} = [\ u[x := t]\ ]$.

- If $u = x$, then $[u]^x_{[t]} = [t]$. On the other side, $[\ u[x := t]\ ] = [t]$.
- If u is a constant/variable distinct from x, then $[u]^x_{[t]} = [u]$. On the other side, $[\ u[x := t]\ ] = [u]$.
- If u has form $f(t_1, \ldots, t_n)$ with $n > 0$, then $[f(t_1, \ldots, t_n)]^x_{[t]} = [f]^x_{[t]}([t_1]^x_{[t]}, \ldots, [t_n]^x_{[t]})$. Using induction, each $[t_i]^x_{[t]} = [\ t_i[x := t]\ ]$. Also $[f]^x_{[t]} = [f]$. As as a consequence,

$$[f]^x_{[t]}([t_1]^x_{[t]}, \ldots, [t_n]^x_{[t]}) = [f]([\ t_1[x := t], \ldots, t_n[x := t]\ ) =$$

$$[\ f(t_1[x := t], \ldots, t_n[x := t]\ )\ ] = [\ f(t_1, \ldots, t_n)[x := t]\ ].$$

We prove Theorem 2 by induction:

- Assume that F is an atom $P(t_1, \ldots, t_n)$. Then $(D, [\ ]^x_{[t]}) \models P(t_1, \ldots, t_n)$ iff $([t_1]^x_{[t]}, \ldots, [t_n]^x_{[t]}] \in [P]^x_{[t]}$. By the term property on the previous page, this is equivalent to $([\ t_1[x := t]\ ], \ldots, [\ t_n[x := t]\ ]) \in [P]$, which in turn is equivalent to $(D, [\ ]) \models P(t_1, \ldots, t_n)[x := t]$.

- The propositional operators are straightforward.

- Assume that F has form $\forall x\ P$. Then $(D, [\ ]) \models \forall x\ P$ iff $(D, [\ ])^x_{[t]} \models \forall x\ P$. At the same time $(\forall x\ P)[x := t] = (\forall x\ P)$, so that $(D, [\ ]) \models (\forall x\ P)[x := t]$.

- Assume that F has form $\forall y\ P$ with $x \neq y$, that x is free in P, but y is not free in t.

  Then $(D, [\ ]^{x}_{[t]}) \models \forall y\ P$ iff for all $d \in D$, $\ (D, [\ ]^{x\ \ y}_{[t]\ d}) \models P$. Because $x \neq y$, this is equivalent to $(D, [\ ]^{y\ \ x}_{d\ [t]}) \models P$. By induction, this is equivalent to: For all $d \in D$, $(D, [\ ]^{y}_{d}) \models P[x := t]$. This again is equivalent to $(D, [\ ]) \models \forall y\ (P[x := t])$.

- Assume that F has form $\forall y\ P$ with $x \neq y$, that x is free in P and that y is free in t.

  Then $(\forall y\ P)[x := t]$ equals $\forall z\ (P[y := z][x := t])$, where z is a variable which does not occur in t and which is not free in P.

  By definition, $(D, [\,]^{x}_{[t]}) \models \forall y\ P$ iff

  for all $d \in D$, $\quad (D, [\,]^{x}_{[t]}\,{}^{y}_{d}) \models P$.

  Because z does not occur in P, this is the case iff

  for all $d \in D$, $\quad (D, [\,]^{x}_{[t]}\,{}^{y}_{d}\,{}^{z}_{d}) \models P \Leftrightarrow (D, [\,]^{x}_{[t]}\,{}^{z}_{d}\,{}^{y}_{d}) \models P$.

  This then is equivalent to:

  for all $d \in D$, $\quad (D, [\,]^{x}_{[t]}\,{}^{z}_{d}\,{}^{y}_{[z]^{x}_{[t]}\,{}^{z}_{d}}) \models P$, which by induction is equivalent to

  for all $d \in D$, $\quad (D, [\,]^{x}_{[t]}\,{}^{z}_{d}) \models P[y := z]$.

  Because $x \neq z$, this is equivalent to:

  For all $d \in D$, $\quad (D, [\,]^{z}_{d}\,{}^{x}_{[t]}) \models P[y := z]$, which is equivalent to $(D, [\,]^{z}_{d}) \models P[y := z][x := t]$, for every D. This, finally, is equivalent to $(D, [\,]) \models \forall z\ (P[y := z][x := t])$.

Soundness of $\forall$-left

Assume that the sequent $\Gamma, P[x := t] \vdash \Delta$ is a valid sequent.

We have to show show that $\Gamma,\ \forall x\ P(x) \vdash \Delta$ is a valid sequent as well. Let $I = (D, [\ ])$ be an arbitrary interpretation which makes all formulas $A \in \Gamma \cup [\forall x\ P(x)]$ true.

Let d be the interpretation of t in D, i.e. $d = [t]$.

From $(D, [\ ]) \models \forall x\ P$, follows that $(D, [\ ]^x_d) \models P$.

From Theorem 2 follows that $(D, [\ ])$ models $P[x := t]$.

From the validity of $\Gamma, P[x := t] \vdash \Delta$ follows that $(D, [\ ])$ makes a formula in Δ true.

The following property is needed, because we allow free replacement of α-equivalent formulas in sequents.

Theorem 3: Let F_1 and F_2 be two formulas, s.t. $F_1 \equiv_\alpha F_2$. Let $(D, [\ ])$ be an interpretation. Then $(D, [\ ]) \models F_1$ iff $(D, [\ ]) \models F_2$.

Soundness of repl

Assume that $\Gamma[x := t_1], t_1 = t_2 \vdash \Delta[x := t_1]$ is a valid sequent. We have to show that $\Gamma[x := t_2], t_1 = t_2 \vdash \Delta[x := t_2]$ is also a valid sequent.

Assume that $(D, [\ ])$ makes all formulas in $\Gamma[x := t_2]$ and $t_1 = t_2$ true. Define $d = [t_1] = [t_2]$.

By Theorem 2, $(D, [\ ]^x_d)$ makes all formulas of Γ true. By (one more time) Theorem 2, $(D, [\ ])$ makes all formulas of $\Gamma[x := t_1]$ true.

It follows that $(D, [\ ])$ makes a formula in $\Delta[x := t_1]$ true. Two more applications of Theorem 2 show us that $(D, [\ ])$ makes a formula in $\Delta[x := t_2]$ true.

Completeness of Sequent Calculus

Completeness (of sequent calculus) means: If a sequent $\Gamma \vdash \Delta$ is valid, then it has a proof. (in sequent calculus)

We prove that: If a sequent $\Gamma \vdash \Delta$ has no proof in sequent calculus, then there is a counter interpretation. (An interpretion in which all Γ are true, and all Δ are false)

For propositional logic, completeness was proven by giving an algorithm that either constructs a counter model, or a proof of a sequent.

For predicate logic, such an algorithm is not possible, because provability in predicate logic is undecidable.

Instead, we give an algorithm which either

1. returns a proof of $\Gamma \vdash \Delta$,
2. or approximates an interpretation $I = (D, [\,])$, in which all Γ are true, and all Δ are false.

The algorithm systematically searches for a counter interpretation.

The algorithm can be simplified if one uses the following normal form:

Definition: A formula F is in negation normal form (NNF) if it does not contain $\leftrightarrow$ or $\rightarrow$, and $\neg$ is applied on atoms only.

$\neg\forall x\ p(x, x)$ is not in NNF, but $\exists x\ \neg p(x, x)$ is.

Similarly, $A \rightarrow B$ is not in NNF, but $\neg A \vee B$ is.

The advantage of the NNF is that it reduces the number of operators.

Theorem: For every formula F, there exists a formula F' s.t. F' is in NNF, and the sequents $F \vdash F'$ and $F' \vdash F$ are provable.

We give an algorithm that constructs F' and the two required proofs. The algorithm is essentially recursive, but one has to be a bit careful with polarity changes.

If one has a formula $A \rightarrow B$, it can be replaced by $\neg A \vee B$. If one has a formula $\neg(A \rightarrow B)$, it cannot be replaced by $\neg(\neg A \vee B)$, because this is not in NNF. Instead, one has to replace $\neg(A \rightarrow B)$ by $A \wedge \neg B$.

The solution for this problem is that we 'count' negations. For this, polarities are used. A subformula has either positive polarity $+$ or negative polarity $-$:

Let F be a formula, let P be a polarity. $\mathrm{NNF}(F, P)$ returns a triple (F', π_1, π_2), where F' is in NNF, and π_1, π_2 are proofs, s.t.

If $\pi = +$, then π_1 is a proof of $F \vdash F'$, and π_2 is a proof of $F' \vdash F$.

If $\pi = -$, then π_1 is a proof of $\vdash F, F'$ and π_2 is a proof of $F, F' \vdash$

Recursive Definition of NNF(F, P) :

- Let A be an atom. NNF$(A, +)$ returns (A, π, π) in which π is the axiom proving $A \vdash A$.

  NNF$(A, -) = (\neg A, \pi_1, \pi_2)$, in which π_1 is a (trivial) proof of $\vdash A, \neg A$, and π_2 is an (also trivial) proof of $\neg A, A \vdash$.

- Assume that F has form $\neg A$. Then NNF$(\neg A, +)$ is obtained as follows: Recursively assign $(F', \pi_1, \pi_2) = \text{NNF}(A, -)$.

  Then we return

$$\left(\quad F', \quad \frac{\pi_1}{\neg A \vdash F'}(\neg\text{-left}), \quad \frac{\pi_2}{F' \vdash \neg A}(\neg\text{-right}) \quad \right)$$

  NNF$(\neg A, -)$ is defined as follows: First recursively assign $(F', \pi_1, \pi_2) = \text{NNF}(A, +)$. Then return

$$\left(\quad F', \quad \frac{\pi_1}{\vdash \neg A, F'}(\neg\text{-right}), \quad \frac{\pi_2}{\neg A, F \vdash}(\neg\text{-left}) \quad \right)$$

- Assume that F has form $A_1 \vee A_2$. Then $\mathrm{NNF}(A_1 \vee A_2, +)$ is computed as follows: Recursively assign
  $(\pi_1, \pi_2, B_1) = \mathrm{NNF}(A_1, +), \quad (\rho_1, \rho_2, B_2) = \mathrm{NNF}(A_2, +).$
  Then we return a triple consisting of:

$$B_1 \vee B_2,$$

$$\frac{\pi_1 \qquad \rho_1}{A_1 \vee A_2 \vdash B_1 \vee B_2}(\vee)\text{-left and right.}$$

$$\frac{\pi_2 \qquad \rho_2}{B_1 \vee B_2 \vdash A_1 \vee A_2}(\vee)\text{-left and right.}$$

- Assume that F has form $A_1 \vee A_2$. Then $\mathrm{NNF}(A_1 \vee A_2, -)$ is computed as follows: Recursively assign
  $(\pi_1, \pi_2, B_1) = \mathrm{NNF}(A_1, -), \quad (\rho_1, \rho_2, B_2) = \mathrm{NNF}(A_2, -).$
  Then we return a triple consisting of:

$$B_1 \wedge B_2,$$

$$\frac{\pi_1 \qquad \rho_1}{\vdash A_1 \vee A_2, B_1 \wedge B_2}(\wedge\text{-right and} \vee\text{-right})$$

$$\frac{\pi_2 \qquad \rho_2}{B_1 \wedge B_2, A_1 \vee A_2 \vdash}(\vee\text{-left and} \wedge\text{-left})$$

- Assume that F has form $\forall x\ A$. Then $\mathrm{NNF}(\forall x\ A, +)$ is computed as follows: First recursively define $(B, \pi_1, \pi_2) = \mathrm{NNF}(A, +)$.

  Then we return

$$\begin{array}{c}(\quad \forall x\ B, \\ \dfrac{\pi_1}{\forall x\ A \vdash \forall x\ B}\ (\forall\text{-left and right}), \\ \dfrac{\pi_2}{\forall x\ B \vdash \forall x\ A}\ (\forall\text{-left and right}) \quad).\end{array}$$

- Assume that F has form $\forall x\ A$. Then $\mathrm{NNF}(\forall x\ A, -)$ is computed as follows: First recrusively define $(B, \pi_1, \pi_2) = \mathrm{NNF}(A, -)$.

  Then we return

$$\begin{array}{c} (\quad \exists x\ B, \\ \dfrac{\pi_1}{\vdash \forall x\ A,\ \exists x\ B}\ (\forall\text{-right and }\exists\text{-right}), \\ \dfrac{\pi_2}{\forall x\ A,\ \exists x\ B \vdash}\ (\forall\text{-right and }\exists\text{-left})\quad). \end{array}$$

The other operators $\wedge, \rightarrow, \leftrightarrow, \exists, \perp, \top$ are variations of the same principles.

In case there are nested $\leftrightarrow$s, the NNF can be exponentially bigger than F. In case there are no $\leftrightarrow$s in F, the resulting NNF has a size linear in F.

Completeness of Sequent Calculus for Formulas in NNF

We show completeness for sequents of the form $A_1, \ldots, A_p \vdash$, in which each A_i is in NNF.

In order to do this, we define a semi-algorithm which searches for an interpretation $(D, [\ ])$ in which all A_i are true.

If $A_1, \ldots, A_p \vdash$ has a proof, then the algorithm will find a proof and terminate. Otherwise, it either finds a counter interpretation in finite time, or it keeps on approximating a counter interpretation forever.

We show that in both cases the algorithm constructs in the limit an interpretation $(D, [\ ])$ which makes all A_i true.

We will later see that provability in first-order logic is undecidable, and that no terminating algorithm is possible.

The semi-algorithm ModSearch($\mathcal{C}, \Gamma$) has two parameters $\mathcal{C}$ and Γ which need fulfill the following requirements:

- $\mathcal{C}$ is a set of constants that do not occur in the original formulas $A_1, \ldots A_m$.
- Γ is a finite set of formulas. It contains at least the following formulas:
  - For each pair of distinct $c_1, c_2 \in \mathcal{C}$ a formula $c_1 \not\approx c_2$.
  - For each function symbol f that occurs in an A_i, a formula $\forall x_1 \cdots x_n \; \exists y \; f(x_1, \ldots, x_n) \approx y$.
  - For each predicate symbol p that occurs in an A_i, a formula $\forall x_1 \cdots x_n \; p(x_1, \ldots, x_n) \vee \neg p(x_1, \ldots, x_n)$.
  - The initial formulas $A_1, \ldots, A_m$.

In addition to the formulas mentioned on the previous slide, Γ may also contain subformulas and instances of subformulas of these formulas. An instance is obtained by assigning constants from $\mathcal{C}$ to a universally or existentially quantified variable.

We show that either ModSearch$(\mathcal{C}, \Gamma)$ constructs a proof of $\Gamma \vdash$ or $\Gamma \vdash$ has a model $(D, [\,])$.

ModSearch tries to construct a model by adding formulas to Γ. On each call, it adds a few formulas to Γ and then recursively calls itself. Either it reaches a Γ that is obviously contradictive, or the algorithm will continue calling itself with bigger and bigger Γ's. We will show that in this case, the growing sequence of Γ's defines a model.

ModSearch($\mathcal{C}, \Gamma$) is initially called with $\mathcal{C} = \{c\}$ and $\Gamma = A_1, \ldots, A_m$.

After that, at each point ModSearch($\mathcal{C}, \Gamma$) can do one of the following things:

- Select a formula of form $A \wedge B$ in Γ for which either $A \not\in \Gamma$ or $B \not\in \Gamma$. After that, call

$$\text{ModSearch}(\ \mathcal{C},\ \Gamma \cup \{A, B\}\ ).$$

- Select a formula of form $A \vee B$ in Γ for which neither $A \in \Gamma$, nor $B \in \Gamma$. If both

$$\text{ModSearch}(\ \mathcal{C},\ \Gamma \cup \{A\}\ ) \text{ and } \text{ModSearch}(\ \mathcal{C},\ \Gamma \cup \{B\}\ )$$

  return a proof, then one can construct a proof of $\Gamma \vdash$.

- Select a universally quantified formula $\forall x\ F$, and a $c \in \mathcal{C}$ for which $F[x := c] \notin \Gamma$ and call

$$\text{ModSearch}(\ \mathcal{C}, \Gamma \cup \{F[x := c]\}\ ).$$

- Select an existentially quantified formula $\exists x\ F$ for which there is no instance $F[x := c]$ in Γ.

  If for $c_i \in \mathcal{C} = \{c_1, \ldots, c_n\}$ all of

$$\text{ModSearch}(\ \mathcal{C}, \Gamma \cup \{F[x := c_i]\}\ ),$$

  and in addition for a new $\hat{c} \notin \mathcal{C} = \{c_1, c_2, \ldots, c_n\}$,

$$\text{ModSearch}(\ \mathcal{C} \cup \{\hat{c}\},$$

$$\Gamma \cup \{\hat{c} \not\approx c_1, \hat{c} \not\approx c_2, \ldots, \hat{c} \not\approx c_n,\ F[x := \hat{c}]\ \}\ )$$

  returns a proof, then one can construct a proof of $\Gamma \vdash$.

- Write Γ in form $\Gamma \cup \{f(c_1, \ldots, c_n) \approx \hat{c}\} \vdash$, where $c_1, \ldots, c_n, \hat{c} \in \mathcal{C}$. Recursively call

  $$\mathrm{ModSearch}(\mathcal{C}, \Gamma[f(c_1, \ldots, c_n) \Rightarrow \hat{c}] \cup \{f(c_1, \ldots, c_n) \approx \hat{c}\} \vdash$$

  If the result is a proof, then one can construct a proof of $\Gamma \cup \{f(c_1, \ldots, c_n) \approx \hat{c}\} \vdash$.

- If Γ contains $\bot$, two complementary literals of form $A, \neg A$, or a disequality $c \not\approx c$, then

  $$\mathrm{ModSearch}(\mathcal{C}, \Gamma \vdash\ )$$

  constructs a proof of $\Gamma \vdash$.

It is assumed that rules are applied in a fair order. That means that every rule that can be applied in a sequence of recursively calls of ModSearch, will eventually will be applied.

In case ModSearch$(\mathcal{C}, \Gamma)$ does constructs a proof, it produces a sequence of recursive calls with arguments

$$(\mathcal{C}_0, \Gamma_0),\ (\mathcal{C}_1, \Gamma_1), \ldots$$

s.t. for each i,

$$\mathcal{C}_i \subseteq \mathcal{C}_{i+1}, \text{ and } \Gamma_i \subseteq \Gamma_{i+1}.$$

This sequence either ends, because no further rule can be applied, or it is infinite. In the first case, define $(\mathcal{C}^*, \Gamma^*) = (\mathcal{C}_n, \Gamma_n)$, for the last n in the sequence.

In the second case, define

$$(\mathcal{C}^*, \Gamma^*) = (\bigcup \mathcal{C}_i, \bigcup \Gamma_i).$$

The pair $(\mathcal{C}^*, \Gamma^*)$ has the following properties:

- Γ does not contain $\perp$.
- If $A \wedge B \in \Gamma^*$ then both $A \in \Gamma^*$ and $B \in \Gamma^*$.
- If $A \vee B \in \Gamma^*$, then either $A \in \Gamma^*$ or $B \in \Gamma^*$.
- If $\forall x\ F$ in Γ^* and $c \in \mathcal{C}$, then $F[x := c] \in \Gamma^*$.
- If $\exists x\ F \in \Gamma^*$ then there is a $c \in \mathcal{C}$ for which $F[x := c] \in \Gamma^*$.
- For each pair $c_1, c_2 \in \mathcal{C}^*$ with $c_1 \neq c_2$, the formula $c_1 \not\approx c_2$ is in Γ^*. For each $c \in \mathcal{C}^*$, the formula $c \not\approx c$ is not in Γ^*.

- For each function symbol f occurring in $\Gamma^* \backslash \mathcal{C}^*$ with arity n, for each n-tuple $(c_1, \ldots, c_n) \in {\mathcal{C}^*}^n$, there is exactly one $\hat{c} \in \mathcal{C}$, s.t. $f(c_1, \ldots, c_n) \approx \hat{c} \ \in \Gamma^*$.
- For each predicate symbol p occurring in Γ^* with arity n, for each n-tuple $(c_1, \ldots, c_n) \in {\mathcal{C}^*}^n$, exactly one of $p(c_1, \ldots, c_n)$ or $\neg p(c_1, \ldots, c_n)$ is in Γ^*.

Let $\mathcal{C}^*$ and Γ^* be sets as defined on the previous slide.

Define the following interpretation $(D, [\ ])$.

- $D = \mathcal{C}$.
- For each $f \in \mathcal{F}^* \backslash \mathcal{C}^*$ with arity n, for each n-tuple $(c_1, \ldots, c_n) \in \mathcal{C}^{*n}$, there is exactly one equality of form $f(c_1, \ldots, c_n) \approx \hat{c}$ in Γ^*. Put $[f](c_1, \ldots, c_n) = \hat{c}$.
- For each $p \in \mathcal{P}$ with arity n, for each n-tuple $(c_1, \ldots, c_n) \in \mathcal{C}^{*n}$, put
$$[p] = \{(c_1, \ldots, c_n) \mid p(c_1, \ldots, c_n) \in \Gamma^*\}.$$

Theorem: For every formula F, $\quad (D, [\ ]) \models F$ iff $F \in \Gamma$.
proof (induction. It turns out that the hardest cases are the atoms and the equalities. Here you have to make an induction on the number of symbols that are not in $\mathcal{C}$)

The Complete Procedure

Suppose we are given sequent $A_1, \ldots, A_p \vdash B_1, \ldots, B_q$.

Write $\Gamma \vdash$ for the sequent $A_1, \ldots, A_p, \neg B_1, \ldots, \neg B_q \vdash$.

Let $\Gamma' \vdash$ be the sequent that one obtains by replacing each $F \in \Gamma$ by its NNF F'.

ModSearch($\{c\}, \Gamma'$) returns either a proof of $\Gamma' \vdash$, or it constructs/approximates a model of Γ'.

If the result is a proof, the proof can be transformed into a proof of $\Gamma \vdash$, using the proofs returned by the NNF algorithm, and a lot of cuts. Using cuts on the sequents $\vdash B_1, \neg B_1, \ldots, \vdash B_q, \neg B_q$, a proof of $A_1, \ldots, A_p \vdash B_1, \ldots, B_q$ can be obtained.

If ModSearch($\{c\}, \Gamma'$) constructs a model of Γ', then this model is also a model of Γ, because each formula in Γ is provable from its NNF in Γ'.

It is easily checked that the model is a counter model of $A_1, \ldots, A_p \vdash B_1, \ldots, B_q$.

Theorem: (Compactness theorem). Let Γ be a set of first-order formulas, not necessarily finite. (but with a well-order) If every $\Gamma' \subseteq \Gamma$ has a model, then Γ also has a model.

proof: Procedure ModSearch also works when Γ is infinite. Because Γ has a well-order, one can define a fair rule application strategy. If ModSearch finds a proof, then this proof is a finite object. Hence it uses a finite subset of Γ.

Non-definability of Natural Numbers

Theorem There exists no set of first-order formulas over a signature containing a constant 0 and a unary function s, s.t. for every interpretation $(D, [\ ])$, $(D, [0], [s])$ is isomorphic to the natural numbers.

Another example of a non-expressible property is reachability in graphs.