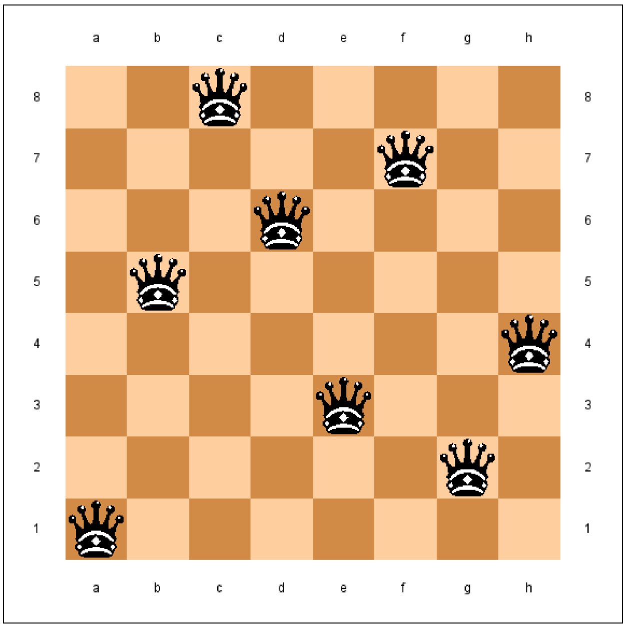
Formalising problems in propositional logic

N-Queens Problem

N-Queens Problem.

Place N queens on an $N \times N$ chess board such that no two queens attack each other.

Next: Formalising N-Queens Problem in propositional logic.

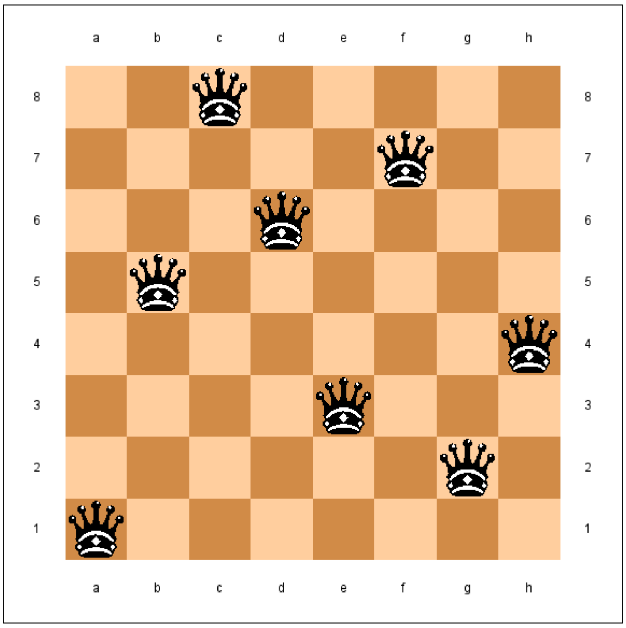


Formalising N-Queens Problem (I)

Propositional variables: q_{ij} – square (i, j) is occupied by a queen.

Rules: If q_{ij} is placed then there should be no other queen placed on

- row right: $(i, j + k)$
  for $1 \leq k \leq n - j$,
- column up: $(i + k, j)$
  for $1 \leq k \leq n - i$
- diag. up right: $(i + k, j + k)$
  for $1 \leq k \leq \min\{n - i, n - j\}$
- diag. up left: $(i + k, j - k)$
  for $1 \leq k \leq \min\{n - i, j - 1\}$

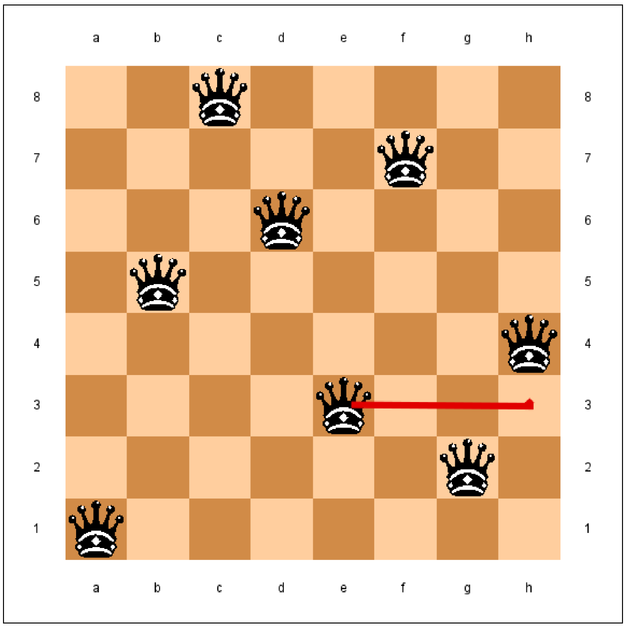


Formalising N-Queens Problem (I)

Propositional variables: q_{ij} – square (i, j) is occupied by a queen.

Rules: If q_{ij} is placed then there should be no other queen placed on

- row right: $(i, j + k)$
  for $1 \leq k \leq n - j$,
- column up: $(i + k, j)$
  for $1 \leq k \leq n - i$
- diag. up right: $(i + k, j + k)$
  for $1 \leq k \leq \min\{n - i, n - j\}$
- diag. up left: $(i + k, j - k)$
  for $1 \leq k \leq \min\{n - i, j - 1\}$

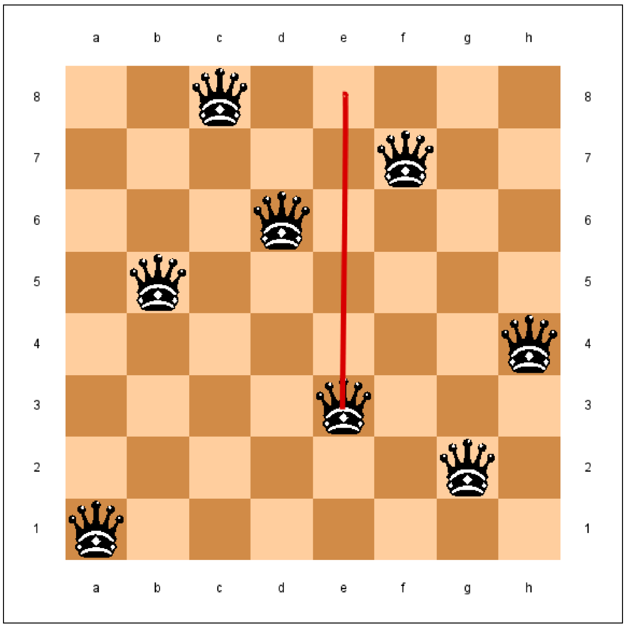


Formalising N-Queens Problem (I)

Propositional variables: q_{ij} – square (i, j) is occupied by a queen.

Rules: If q_{ij} is placed then there should be no other queen placed on

- row right: $(i, j + k)$
  for $1 \leq k \leq n - j$,
- column up: $(i + k, j)$
  for $1 \leq k \leq n - i$
- diag. up right: $(i + k, j + k)$
  for $1 \leq k \leq \min\{n - i, n - j\}$
- diag. up left: $(i + k, j - k)$
  for $1 \leq k \leq \min\{n - i, j - 1\}$

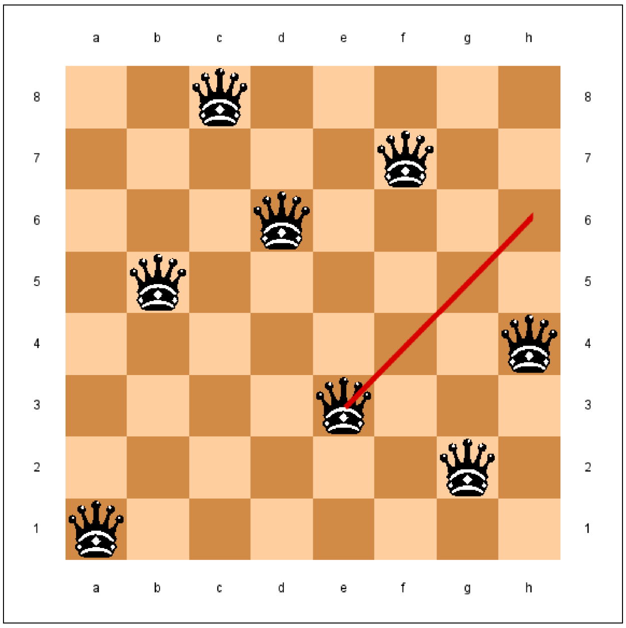


Formalising N-Queens Problem (I)

Propositional variables: q_{ij} – square (i, j) is occupied by a queen.

Rules: If q_{ij} is placed then there should be no other queen placed on

- row right: $(i, j + k)$
  for $1 \leq k \leq n - j$,
- column up: $(i + k, j)$
  for $1 \leq k \leq n - i$
- diag. up right: $(i + k, j + k)$
  for $1 \leq k \leq \min\{n - i, n - j\}$
- diag. up left: $(i + k, j - k)$
  for $1 \leq k \leq \min\{n - i, j - 1\}$

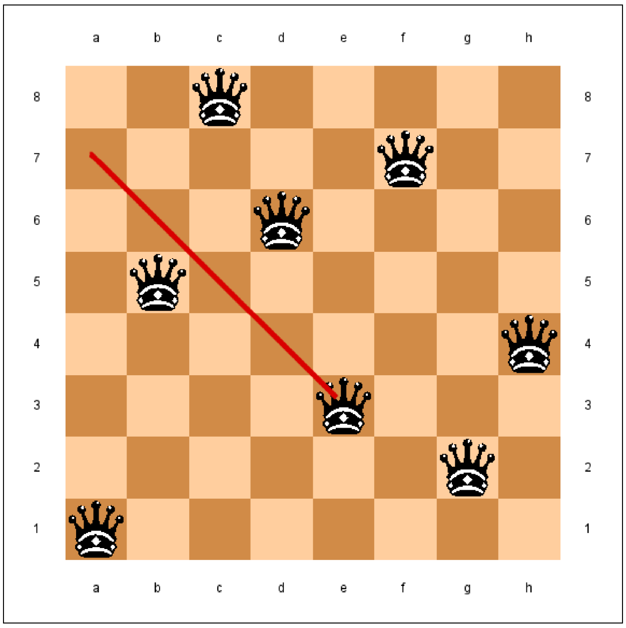


Formalising N-Queens Problem (I)

Propositional variables: q_{ij} – square (i, j) is occupied by a queen.

Rules: If q_{ij} is placed then there should be no other queen placed on

- row right: $(i, j + k)$
  for $1 \leq k \leq n - j$,
- column up: $(i + k, j)$
  for $1 \leq k \leq n - i$
- diag. up right: $(i + k, j + k)$
  for $1 \leq k \leq \min\{n - i, n - j\}$
- diag. up left: $(i + k, j - k)$
  for $1 \leq k \leq \min\{n - i, j - 1\}$



Formalising N-Queens Problem (I)

Propositional variables: q_{ij} – square (i, j) is occupied by a queen.

Rules: If q_{ij} is placed then there should be no other queen placed on

- row right: $(i, j + k)$
  for $1 \leq k \leq n - j$,
- column up: $(i + k, j)$
  for $1 \leq k \leq n - i$
- diag. up right: $(i + k, j + k)$
  for $1 \leq k \leq \min\{n - i, n - j\}$
- diag. up left: $(i + k, j - k)$
  for $1 \leq k \leq \min\{n - i, j - 1\}$

- $R_{ij} = \bigwedge_k (q_{ij} \rightarrow \neg q_{i,j+k})$
  for $1 \leq k \leq n - j$,
- $C_{ij} = \bigwedge_k (q_{ij} \rightarrow \neg q_{i+k,j})$
  for $1 \leq k \leq n - i$
- $\mathrm{DRU}_{ij} = \bigwedge_k (q_{ij} \rightarrow \neg q_{i+k,j+k})$
  for $1 \leq k \leq \min\{n - i, n - j\}$
- $\mathrm{DLU}_{ij} = \bigwedge_k (q_{ij} \rightarrow \neg q_{i-k,j+k})$
  for $1 \leq k \leq \min\{n - i, j - 1\}$

Formalising N-Queens Problem (I)

Propositional variables: q_{ij} – square (i, j) is occupied by a queen.

Rules: If q_{ij} is placed then there should be no other queen placed on

- row right: $(i, j+k)$ for $1 \le k \le n-j$,
- column up: $(i+k, j)$ for $1 \le k \le n-i$
- diag. up right: $(i+k, j+k)$ for $1 \le k \le \min\{n-i, n-j\}$
- diag. up left: $(i+k, j-k)$ for $1 \le k \le \min\{n-i, j-1\}$

- $R_{ij} = \bigwedge_k (q_{ij} \to \neg q_{i,j+k})$ for $1 \le k \le n-j$,
- $C_{ij} = \bigwedge_k (q_{ij} \to \neg q_{i+k,j})$ for $1 \le k \le n-i$
- $\mathrm{DRU}_{ij} = \bigwedge_k (q_{ij} \to \neg q_{i+k,j+k})$ for $1 \le k \le \min\{n-i, n-j\}$
- $\mathrm{DLU}_{ij} = \bigwedge_k (q_{ij} \to \neg q_{i-k,j+k})$ for $1 \le k \le \min\{n-i, j-1\}$

Formalising N-Queens Problem (I)

Propositional variables: q_{ij} – square (i, j) is occupied by a queen.

Rules: If q_{ij} is placed then there should be no other queen placed on

- row right: $(i, j+k)$ for $1 \le k \le n-j$,
- column up: $(i+k, j)$ for $1 \le k \le n-i$
- diag. up right: $(i+k, j+k)$ for $1 \le k \le \min\{n-i, n-j\}$
- diag. up left: $(i+k, j-k)$ for $1 \le k \le \min\{n-i, j-1\}$

- $R_{ij} = \bigwedge_k (q_{ij} \to \neg q_{i,j+k})$ for $1 \le k \le n-j$,
- $C_{ij} = \bigwedge_k (q_{ij} \to \neg q_{i+k,j})$ for $1 \le k \le n-i$
- $\mathrm{DRU}_{ij} = \bigwedge_k (q_{ij} \to \neg q_{i+k,j+k})$ for $1 \le k \le \min\{n-i, n-j\}$
- $\mathrm{DLU}_{ij} = \bigwedge_k (q_{ij} \to \neg q_{i-k,j+k})$ for $1 \le k \le \min\{n-i, j-1\}$

Formalising N-Queens Problem (I)

Propositional variables: q_{ij} – square (i,j) is occupied by a queen.

Rules: If q_{ij} is placed then there should be no other queen placed on

- row right: $(i, j+k)$ for $1 \leq k \leq n-j$,
- column up: $(i+k, j)$ for $1 \leq k \leq n-i$
- diag. up right: $(i+k, j+k)$ for $1 \leq k \leq \min\{n-i, n-j\}$
- diag. up left: $(i+k, j-k)$ for $1 \leq k \leq \min\{n-i, j-1\}$

- $R_{ij} = \bigwedge_k (q_{ij} \to \neg q_{i,j+k})$ for $1 \leq k \leq n-j$,
- $C_{ij} = \bigwedge_k (q_{ij} \to \neg q_{i+k,j})$ for $1 \leq k \leq n-i$
- $\mathrm{DRU}_{ij} = \bigwedge_k (q_{ij} \to \neg q_{i+k,j+k})$ for $1 \leq k \leq \min\{n-i, n-j\}$
- $\mathrm{DLU}_{ij} = \bigwedge_k (q_{ij} \to \neg q_{i-k,j+k})$ for $1 \leq k \leq \min\{n-i, j-1\}$

Formalising N-Queens Problem (II)

$$\begin{aligned}
\mathrm{QueenRules} &= \bigwedge_{ij}(R_{ij} \wedge C_{ij} \wedge \mathrm{DRU}_{ij} \wedge \mathrm{DLU}_{ij}) \\
\mathrm{QueenPlaced}_i &= q_{i1} \vee \ldots \vee q_{in} \\
\mathrm{NQueensPlaced} &= \bigwedge_i \mathrm{QueenPlaced}_i \\
\mathrm{NQueensProblem} &= \mathrm{QueenRules} \bigwedge \mathrm{NQueensPlaced}
\end{aligned}$$

Lemma

N-Queens Problem has a solution if and only if NQueensProblem is satisfiable.

Formalising N-Queens Problem (II)

$$
\begin{aligned}
\text{QueenRules} &= \bigwedge_{ij}(R_{ij} \wedge C_{ij} \wedge \text{DRU}_{ij} \wedge \text{DLU}_{ij}) \\
\text{QueenPlaced}_i &= q_{i1} \vee \ldots \vee q_{in} \\
\text{NQueensPlaced} &= \bigwedge_i \text{QueenPlaced}_i \\
\text{NQueensProblem} &= \text{QueenRules} \bigwedge \text{NQueensPlaced}
\end{aligned}
$$

Lemma

N-Queens Problem has a solution if and only if NQueensProblem *is satisfiable.*

Formalising N-Queens Problem (II)

$$
\begin{aligned}
\text{QueenRules} &= \bigwedge_{ij}(R_{ij} \wedge C_{ij} \wedge \text{DRU}_{ij} \wedge \text{DLU}_{ij}) \\
\text{QueenPlaced}_i &= q_{i1} \vee \ldots \vee q_{in} \\
\text{NQueensPlaced} &= \bigwedge_i \text{QueenPlaced}_i \\
\text{NQueensProblem} &= \text{QueenRules} \bigwedge \text{NQueensPlaced}
\end{aligned}
$$

Lemma

N-Queens Problem has a solution if and only if NQueensProblem *is satisfiable.*

Formalising N-Queens Problem (II)

$$
\begin{aligned}
\mathrm{QueenRules} &= \bigwedge_{ij}(R_{ij} \wedge C_{ij} \wedge \mathrm{DRU}_{ij} \wedge \mathrm{DLU}_{ij}) \\
\mathrm{QueenPlaced}_i &= q_{i1} \vee \ldots \vee q_{in} \\
\mathrm{NQueensPlaced} &= \bigwedge_i \mathrm{QueenPlaced}_i \\
\mathrm{NQueensProblem} &= \mathrm{QueenRules} \bigwedge \mathrm{NQueensPlaced}
\end{aligned}
$$

Lemma

N-Queens Problem has a solution if and only if $\mathrm{NQueensProblem}$ *is satisfiable.*

Formalising N-Queens Problem (II)

$$
\begin{aligned}
\mathrm{QueenRules} &= \bigwedge_{ij}(R_{ij} \wedge C_{ij} \wedge \mathrm{DRU}_{ij} \wedge \mathrm{DLU}_{ij}) \\
\mathrm{QueenPlaced}_i &= q_{i1} \vee \ldots \vee q_{in} \\
\mathrm{NQueensPlaced} &= \bigwedge_i \mathrm{QueenPlaced}_i \\
\mathrm{NQueensProblem} &= \mathrm{QueenRules} \bigwedge \mathrm{NQueensPlaced}
\end{aligned}
$$

Lemma

N-Queens Problem has a solution if and only if $\mathrm{NQueensProblem}$ *is satisfiable.*

Previously . . . (what I expect you to know)

- propositional logic:
  - syntax: propositional symbols
    logical operators $\neg$, $\vee$, $\wedge$, $\rightarrow$, $\leftrightarrow$, $\top$, $\bot$
  - semantics: truth tables, truth values $\mathbf{1}$, $\mathbf{0}$
- conversion to clausal form
  - atoms, literals, clauses (= multi-sets)
- resolution calculus: resolution rule
  factoring rule
- tautology deletion, subsumption deletion
- multi-set extension ordering

First-order logic

Extension of propositional logic
with quantification over individuals

Most important 'unifying' formal logic system

- knowledge representation and reasoning in CS and AI
- specification and verification in software engineering
- semantics of programming languages
- SQL querying language of data bases
- rules in expert systems
- foundation for logic programming

Very expressive

Deficiencies of propositional logic

- In propositional logic we can make statements about truth of propositions

"grass is green", "1 is an even number",
"grass is green" $\wedge \neg$("1 is an even number")

- But not about objects of structures and relations among several objects. E.g.

"If b lies between a and c, then b also lies between c and a"

is a true statement, but $F \to G$ is not an adequate representation

- Propositional logic cannot represent, e.g.,

"*Every* student wants a job", "*Some* students don't like Logic"

What are the new concepts in first-order logic?

- First-order logic = extension of propositional logic
- Statements about objects of structures can be expressed

$$Between(b, a, c) \to Between(b, c, a) \qquad Even(1)$$

- Symbols denoting functions and constants are allowed

$$b,\ a,\ c \qquad 1 \qquad ann \qquad father\_of(ann) \qquad -1 \qquad 1+5$$

- Abstract, schematic statements via variables

$$x+y \qquad Even(x) \qquad Student(x) \qquad Between(0, -x, x)$$

...and quantification ($\exists$ = 'there exists', $\forall$ = 'for all')

$$\exists x\, Even(x) \qquad \forall x\big(Student(x) \to Want\_job(x)\big)$$

Alphabet of a first-order language

- Logical symbols (domain-independent, fixed):
  logical connectives, quantifiers, variables, auxiliary symbols
- Non-logical symbols (domain-specific, flexible):
  function symbols, constants, predicate symbols

Logical symbols in the alphabet

- Logical connectives:

$\bot$	falsehood	$\top$	truth		
$\neg$	not	$\wedge$	and	$\vee$	or
$\rightarrow$	implication	$\leftrightarrow$	equivalence		
$\forall$	for all	$\exists$	there exists	(quantifiers)	
$\approx$	equality symbol				

- Variables: A countably infinite set $\mathcal{X}$ of symbols $x_0, x_1, x_2, \ldots$
  We also use the symbols x, y, z to denote variables
- Auxiliary symbols: ( ) [ ] .

Note: $\approx$ is equality in FOL language; $=$ is syntactic equality

Non-logical symbols in the alphabet

- Function symbols: A finite or countably infinite set $\mathcal{F}$ of symbols f with arity $n \geq 0$, written f/n,
  - If $n = 0$ then f is also called a constant (symbol).

  Notation: f, g, h for function symbols a, b, c for constants
- Predicate symbols: A finite or countably infinite set $\mathcal{P}$ of symbols P with arity $m \geq 0$, written P/m.
  - If $m = 0$ then P is also called a propositional symbol.

  Notation: P, Q, R for pred. symbols; p, q, r for prop. symbols
- We refer to $\Sigma = (\mathcal{F}, \mathcal{P})$ as the signature.

Each function/predicate symbol has an arity which indicates the number of arguments it takes

Arity	0	1	2	3	n
Symbol	nullary	unary	binary	ternary	n-ary

Non-logical symbols for two sample domains

- Constants, functions, propositional symbols and predicate symbols are domain-specific symbols
  - Are flexibly chosen as appropriate for an application
  - Their interpretations are flexible
- Symbols for number theory:

  Constants: $0, 1, 5$
  Functions: s , $-$, $*$, $+$
  Predicates: $<$, $Even$, $Prime$
- Symbols for a student domain

  Constants: ann, 60332
  Functions: $grade$
  Predicates: $Student$, $Likes$, $<$

Terms

- Terms over Σ and $\mathcal{X}$ are formed according to this syntactic rule:

$$\begin{array}{llll} s, t, u & \longrightarrow & x & \text{(first-order variable)} \\ & \mid & a & \text{(constant)} \\ & \mid & f(s_1, \ldots, s_n), \quad n > 0 & \text{(functional term)} \end{array}$$

  where $x \in \mathcal{X}$, $a/0 \in \mathcal{F}$ and $f/n \in \mathcal{F}$

- Examples, in the number theory domain:

$$0 \qquad 1 \qquad 5 \qquad -(1) \qquad *(1, 5) \qquad x \qquad *(x, y)$$

- By $T_\Sigma(\mathcal{X})$ we denote the set of Σ-terms (over $\mathcal{X}$).
- A term not containing any variable is called a ground term.
  By T_Σ we denote the set of ground Σ-terms.

Subterms

- If s is a term and s occurs as a part of another term t, then s is a subterm of t.
- Example:

$$+(+(x, y), s(0))$$

  Subterms:

$$0 \qquad x \qquad s(0) \qquad +(x, y) \qquad +(+(x, y), s(0))$$

  (Are there more?)
- Alternative infix notation for the term: $(x + y) + s(0)$

Atomic formulae

- Atomic formulae over Σ are formed according to:

$$
\begin{array}{lcll}
A, B & \longrightarrow & P(s_1, \ldots, s_n), \quad n \geq 0 & \text{(non-equational atom)} \\
 & | & s \approx t & \text{(equational atom)}
\end{array}
$$

  where $P/n \in \mathcal{P}$

- Atomic formulae are also called atoms
- Examples of atoms are:

$$
Even(x), \qquad P(a, x), \qquad < (0, s(0)), \qquad s(x) \approx y
$$

Aside: Whenever we admit equations as atomic formulae we are in the realm of first-order logic with equality. Admitting equality does not really increase the expressiveness of first-order logic. But deductive systems where equality is treated specifically can be much more efficient.

Formulae of a first-order language

- Formulae over Σ are formed according to:

$$
\begin{array}{lcll}
F, G, H & \longrightarrow & \bot \mid \top & \text{(logical constants)} \\
 & | & A & \text{(atomic formula)} \\
 & | & \neg F & \text{(not)} \\
 & | & (F \star G) & \star \in \{\wedge, \vee, \rightarrow, \leftrightarrow\} \text{ (binary conn.)} \\
 & | & \forall x F & \text{(universal quantification)} \\
 & | & \exists x F & \text{(existentially quantification)}
\end{array}
$$

- $\mathrm{F}_\Sigma(\mathcal{X})$ denotes the set of all first-order formulae over Σ and $\mathcal{X}$
- Formulae without variables are ground.

Formulae in number theory, informal meaning in $\mathbb{N}$

- $Even(1)$
- $<(1, 5)$ in infix form: $1 < 5$
- $Even(x)$
- $0 < x$
- $\exists x. Even(x)$

  There is an even number
- $\forall x. \exists y. x < y$

  For every number x there is a number y greater than x
- $\forall x. Even(*(x, x))$

  Every square number is even
- $\exists x. (Even(x) \wedge Prime(x) \wedge 0 < x)$

  There is an even prime number greater than 0
- $\forall x. \forall y. (x < y \to \neg(y < x))$

  For any x, y, if x is less than y then y is not less than x

From English to first-order logic

Assume the student domain and these symbols are at our disposal

Constants:	ann	for person Ann
	60332	for this course
Functions:	$grade(x, y)$	for the grade of student x in course y
Predicates:	$Student(x)$	for x is a student
	$Likes(x, y)$	for x likes y
	$x < y$	for $x < y$

- Ann is a student
- Some students don't like 60332
- Ann is the student with the best grade in 60332

Subformulae

- If F is a formula and F occurs as a part of another formula G, then F is a subformula of G.
- Example:

$$\forall x \forall y \big(\leq(x, y) \rightarrow \exists z(+(x, z) \approx y) \big)$$

Subformulae

$\leq(x, y)$ $\quad$ $+(x, z) \approx y$ $\quad$ $\exists z(+(x, z) \approx y)$

$\leq(x, y) \rightarrow \exists z(+(x, z) \approx y)$ $\quad$ $\forall y \big(\leq(x, y) \rightarrow \exists z(+(x, z) \approx y) \big)$

(Are there more?)

Using brackets and notation

- We omit brackets according to the following criteria:
  - Binding precedences (from highest to lowest):

$$\neg \qquad \forall x \qquad \exists x$$

$$\vee \qquad \wedge$$

$$\rightarrow \qquad \leftrightarrow$$

  - $\vee$ and $\wedge$ are associative

$$\begin{array}{rl} P \wedge Q \rightarrow R & \text{for} \\ \exists x\, P(x) \wedge \forall y\, \neg Q(y) & \text{for} \\ P \vee Q \vee R & \text{for} \end{array}$$

- Useful tip: When in doubt use brackets

Free & bound variables

- A quantified formula has the form $\mathcal{Q}xF$, where $\mathcal{Q} \in \{\exists, \forall\}$. x is the quantified variable and F is the scope of the quantifier $\mathcal{Q}x$.
- An *occurrence* of a variable x is bound, if it is inside the scope of a quantifier $\mathcal{Q}x$. Otherwise, it is free.

Examples of free and bound variables

$$\forall x \ \overbrace{\big(P(x) \wedge (\exists y \overbrace{P(y)}^{\text{scope of } \exists y} \rightarrow \ Q(y, x))\big)}^{\text{scope of } \forall x}$$

- The two occurrences of x are both bound, as is the first occurrence of y. The second occurrence of y is free.
- Thus a variable may occur both free and bound in a formula.

$$\forall x \ \overbrace{\big(P(x) \wedge \exists y \ \overbrace{(\exists y \overbrace{P(y)}^{\text{scope of } \exists y} \rightarrow \ Q(y, x))}^{\text{scope of } \exists y} \ \big)}^{\text{scope of } \forall x}$$

Open and closed formulae, universal closure

- Formulae without free variables are closed formulae.
  Formulae with at least one free occurrence of a variable are open formulae.

- If $x_1, \ldots, x_n$ are the free variables in F then $\forall F$ denotes the univeral closure of F. That is,

$$\forall F = \forall x_1 \ldots \forall x_n \, F.$$

  E.g., if $F = P(a, x)$ then

$$\forall F = \forall x \, P(a, x).$$

Summary

- syntax of first-order logic
  - logical symbols (fixed): logical connectives, variables
  - non-logical symbols (flexible, depends on application):
    signature = function symbols + predicate symbols
  - terms
  - atoms
  - formulae
- their informal meaning
- subterms, subformulae
- convention for omitting brackets
- scope of a quantifier, free and bound occurrences of variables

Substitution of terms for variables via an example

- Substitution is an operation on terms and formulae
- Substituting terms for variables means simultaneously and independently replacing the variables
- Example:

$$F = P(g(x), y, x)$$

Substituting a for x gives:

$$F\{x/a\} = P(g(a), y, a)$$
$$F\{x/a,\ y/b\} =$$
$$F\{x/a,\ y/f(z)\} =$$

Substitution of terms for variables

- Formally, a substitution is a function $\boldsymbol{\sigma} : \boldsymbol{\mathcal{X}} \rightarrow \mathrm{T}_{\Sigma}(\boldsymbol{\mathcal{X}})$ such that the set

$$\mathrm{Dom}(\boldsymbol{\sigma}) =^{\mathrm{def}} \{x \in \boldsymbol{\mathcal{X}} \mid \boldsymbol{\sigma}(x) \neq x\} \quad \text{is finite.}$$

- $\mathrm{Dom}(\boldsymbol{\sigma})$ is called the domain of $\boldsymbol{\sigma}$.
- $\mathrm{Cod}(\boldsymbol{\sigma}) =^{\mathrm{def}} \{\boldsymbol{\sigma}(x) \mid x \in \mathrm{Dom}(\boldsymbol{\sigma})\}$ is the codomain of $\boldsymbol{\sigma}$.
- The identity substitution, denoted by ϵ, is the (unique) substitution such that $\mathrm{Dom}(\epsilon) = \emptyset$. I.e., for every $x \in \boldsymbol{\mathcal{X}}$, $\epsilon(x) = x$.

Substitutions

- Substitutions are often written $\{x_1/s_1, \ldots, x_n/s_n\}$, where the x_i are pairwise distinct, and defined by:

$$\{x_1/s_1, \ldots, x_n/s_n\}(y) =^{\text{def}} \begin{cases} s_i, & \text{if } y = x_i \\ y, & \text{otherwise} \end{cases}$$

- The modification $\boldsymbol{\sigma}[x \mapsto t]$ of a substitution $\boldsymbol{\sigma}$ at x is defined like $\boldsymbol{\sigma}$ but it substitutes t for x.

Formally: $$\boldsymbol{\sigma}[x \mapsto t](y) =^{\text{def}} \begin{cases} t, & \text{if } y = x \\ \boldsymbol{\sigma}(y), & \text{otherwise} \end{cases}$$

- Alternative notation: $y\boldsymbol{\sigma}$ for $\boldsymbol{\sigma}(y)$, and
  $y\boldsymbol{\sigma}[x \mapsto t]$ for $\boldsymbol{\sigma}[x \mapsto t](y)$.

Application of a substitution to a term

- Let $\boldsymbol{\sigma}$ be a substitution. For each term t the substitution instance $t\boldsymbol{\sigma}$ is inductively defined by:

$$\begin{aligned} x\boldsymbol{\sigma} &= \boldsymbol{\sigma}(x) \\ a\boldsymbol{\sigma} &= a \\ f(s_1, \ldots, s_n)\boldsymbol{\sigma} &= f(s_1\boldsymbol{\sigma}, \ldots, s_n\boldsymbol{\sigma}) \end{aligned}$$

- Note: $x\boldsymbol{\sigma} = x$, if $x \notin \mathrm{Dom}(\boldsymbol{\sigma})$

Important restriction for formulae

- In formulae we want to substitute a term only for a free variable.
  Example: $F = \exists x(x > y)$

$$F\{y/2\} = \exists x(x > 2)$$
$$F\{y/z\} = \exists x(x > z)$$
$$F\{y/x\} = \exists x(x > x) \qquad \text{*trouble*}$$

  y with x substituted for it becomes bound $\rightsquigarrow$ not useful

- More precisely, we don't want any variables in the substituting terms to be captured by a quantifier in the formula.
- Hence the captured variable must be renamed into a "fresh", that is, previously unused, variable z.

Application of a substitution to a formula

- For each formula F the substitution instance $F\sigma$ is inductively defined by:

$$\bot\sigma = \bot \qquad \top\sigma = \top$$
$$P(s_1, \ldots, s_n)\sigma = P(s_1\sigma, \ldots, s_n\sigma)$$
$$(u \approx v)\sigma = (u\sigma \approx v\sigma)$$
$$(\neg F)\sigma = \neg(F\sigma)$$
$$(F \star G)\sigma = (F\sigma \star G\sigma) \qquad \text{for each binary connective } \star$$
$$(\mathcal{Q}x\, F)\sigma = \mathcal{Q}z\,(F\sigma[x \mapsto z]) \qquad \text{with } z \text{ a fresh variable}$$

- We say $E\sigma$ is formed by applying σ to E, where E is an expression (a term or formula).

Important notes

- Applying a substitution does not change constants, function symbols, predicate symbols or logical connectives.
- Components in terms/formulae are changed simultaneously and independently by σ.
- Every substitution is completely determined by its effect on variables.

Formal semantics of first-order logic

- Aim: Give formal definition of the semantics of terms and formulae of FOL.
  Goes back to Tarski.
- FOL is two-valued: As in propositional logic, the truth values are

  1 ("true") **0** ("false")
- Semantics of $\wedge$, $\vee$, $\neg$, ... are the same as in propositional logic
- We also need to:
  - interpret function symbols as functions,
  - interpret predicate symbols as predicates/relations,
  - fix some domain of elements over which these are defined,
  - constants are interpreted as elements of the domain

Interpretation of constant, function & predicate symbols

Idea: We always assume there is a particular non-empty set of objects: the domain of interpretation. The constants, function symbols and predicate symbols are interpreted over this domain.

- Let F be a formula expressed over the signature $\Sigma = (\mathcal{F}, \mathcal{P})$.
- An interpretation for F (over Σ) is a pair

$$\mathcal{I} = (U, \cdot^{\mathcal{I}}), \quad \text{where}$$

  - U is a non-empty set, called the domain of $\mathcal{I}$, and
  - $\cdot^{\mathcal{I}}$ is a function that maps
    1. each constant to an element of U,
    2. each function symbol to a function on U, and
    3. each predicate symbol to a relation on U.

Examples

- Interpretations of the formula

$$\forall x\, P(a, x)$$

  can be:

$U = \mathbb{N}$	$U = \mathbb{N}$	$U = \mathbb{N}$
$a^{\mathcal{I}} = 0$	$a^{\mathcal{I}} = 3$	$a^{\mathcal{I}} = 0$
$P^{\mathcal{I}} = \leq$	$P^{\mathcal{I}} = \leq$	$P^{\mathcal{I}} = >$

  where $\mathbb{N} = \{0, 1, 2, \ldots\}$ is the set of natural numbers.

- In the first case $\forall x\, P(a, x)$ is interpreted as

$$\text{for each } n \in \mathbb{N}, \ \ (0 \leq n)$$

Interpretation of variables

- Intuitively, the interpretation of variables ranges over the elements of the domain U.
  - If the domain is $\mathbb{N}$:
    $x < y$ is interpreted as $\mathbf{1}$, if $x \mapsto 1$ and $y \mapsto 2$
    $x < y$ is interpreted as $\mathbf{0}$, if $x \mapsto 3$ and $y \mapsto 2$
  - We want to interpret $\forall x$ as 'for all elements x in U' and
    $\exists x$ as 'there is an element x in U'.
- A variable assignment relative to $\mathcal{I} = (U, \cdot^{\mathcal{I}})$ is a function

$$\beta : \mathcal{X} \to U$$

  that assigns the variables in $\mathcal{X}$ to elements of the domain.
- Define $\beta[x \mapsto a]$ to be the assignment that is the same as β except that x is mapped to a.

Interpretation of terms

- Let $\mathcal{I}$ be an interpretation and β a variable assignment.
  A term assignment is a function $\mathcal{I}_\beta : \mathrm{T}_\Sigma(\mathcal{X}) \to U$ defined by

$$\begin{aligned} \mathcal{I}_\beta(x) &= \beta(x) \quad \text{for } x \in \mathcal{X} \\ \mathcal{I}_\beta(a) &= a^{\mathcal{I}} \\ \mathcal{I}_\beta(f(s_1, \ldots, s_n)) &= f^{\mathcal{I}}(\mathcal{I}_\beta(s_1), \ldots, \mathcal{I}_\beta(s_n)), \quad n > 0 \end{aligned}$$

- That is, for each term s its meaning under the interpretation $\mathcal{I}$ and variable assignment β is given by $\mathcal{I}_\beta(s)$.
- $\mathcal{I}_\beta[x \mapsto a]$ is defined to be the assignment $\mathcal{I}_{\beta[x \mapsto a]}$.
- Next we extend $\mathcal{I}_\beta$ to formulae.

Interpretation of formulae

As a formula assignment $\mathcal{I}_\beta : \mathrm{F}_\Sigma(\mathcal{X}) \to \{\mathbf{1}, \mathbf{0}\}$ is defined by:

$$\mathcal{I}_\beta(\bot) = \mathbf{0} \qquad \mathcal{I}_\beta(\top) = \mathbf{1}$$

$$\begin{aligned}
\mathcal{I}_\beta(P(s_1, \ldots, s_n)) = \mathbf{1} &\quad \text{iff} \quad P^{\mathcal{I}}(\mathcal{I}_\beta(s_1), \ldots, \mathcal{I}_\beta(s_n)) = \mathbf{1} \\
\mathcal{I}_\beta(s \approx t) = \mathbf{1} &\quad \text{iff} \quad \mathcal{I}_\beta(s) = \mathcal{I}_\beta(t) \\
\mathcal{I}_\beta(\neg F) = \mathbf{1} &\quad \text{iff} \quad \mathcal{I}_\beta(F) = \mathbf{0} \\
\mathcal{I}_\beta(F \wedge G) = \mathbf{1} &\quad \text{iff} \quad \mathcal{I}_\beta(F) = \mathbf{1} \text{ and } \mathcal{I}_\beta(G) = \mathbf{1} \\
\mathcal{I}_\beta(F \vee G) = \mathbf{1} &\quad \text{iff} \quad \mathcal{I}_\beta(F) = \mathbf{1} \text{ or } \mathcal{I}_\beta(G) = \mathbf{1} \\
\mathcal{I}_\beta(F \rightarrow G) = \mathbf{1} &\quad \text{iff} \quad \mathcal{I}_\beta(F) = \mathbf{1} \text{ implies } \mathcal{I}_\beta(G) = \mathbf{1} \\
\mathcal{I}_\beta(F \leftrightarrow G) = \mathbf{1} &\quad \text{iff} \quad \mathcal{I}_\beta(F) = \mathbf{1} \text{ iff } \mathcal{I}_\beta(G) = \mathbf{1} \\
\mathcal{I}_\beta(\forall x F) = \mathbf{1} &\quad \text{iff} \quad \mathcal{I}_{\beta[x \mapsto u]}(F) = \mathbf{1}, \text{ for all } u \in U \\
\mathcal{I}_\beta(\exists x F) = \mathbf{1} &\quad \text{iff} \quad \mathcal{I}_{\beta[x \mapsto u]}(F) = \mathbf{1}, \text{ for some } u \in U
\end{aligned}$$

Examples

- Determine the truth or falsity of $\forall x\, P(a, x)$ in these two interpretations:

$$U = \mathbb{N} \qquad\qquad U = \mathbb{N}$$
$$a^{\mathcal{I}} = 0 \qquad\qquad a^{\mathcal{I}'} = 1$$
$$P^{\mathcal{I}} = \leq \qquad\qquad P^{\mathcal{I}'} = \leq$$

- In the first case,

$$\mathcal{I}_\beta(\forall x\, P(a, x)) = \mathbf{1}, \quad \text{because for all } n \in \mathbb{N},\ 0 \leq n.$$

(We have $0 \leq x$ is true under $\beta[x \mapsto 0]$, $\beta[x \mapsto 1]$, $\beta[x \mapsto 2]$, etc.)

- In the second case,

$$\mathcal{I}'_\beta(\forall x\, P(a, x)) =$$

Satisfiability, models and validity

- F is true in $\mathcal{I}$ under assignment β, if $\mathcal{I}_\beta(F) = \mathbf{1}$.
- F is satisfiable, if for some interpretation $\mathcal{I}$ and some assignment β, $\mathcal{I}_\beta(F) = \mathbf{1}$.
  Otherwise, F is unsatisfiable. Notation: $F \models \bot$
- F is true in $\mathcal{I}$, if for *every* assignment β, $\mathcal{I}_\beta(F) = \mathbf{1}$.
  We also say $\mathcal{I}$ is a model of F. Notation: $\mathcal{I} \models F$.
- F is valid, if for every interpretation $\mathcal{I}$, $\mathcal{I} \models F$. Notation: $\models F$.
- $\mathcal{I}$ is a model for a set N of formulae, if every formula in N is true in $\mathcal{I}$. Notation: $\mathcal{I} \models N$.

Semantic entailment and equivalence

Let N be a set of formulae, and F and G formulae.

- N entails F, or F semantically follows from N, if every model of N is also a model of F. Notation: $N \models F$.
- We write $G \models F$ instead of $\{G\} \models F$ and
  $\models F$ instead of $\{\} \models F$.
- F and G are semantically equivalent, if $F \models G$ and $G \models F$.
  Notation: $F \equiv G$.

Entailment, equivalence and validity

- Let $N = \{F_1, \ldots, F_n\}$. For *closed* formulae:

  Property 1

  1. $F \equiv G$ iff $\models F \leftrightarrow G$ (F and G are equiv. iff $F \leftrightarrow G$ is valid)
  2. $F \models G$ iff $\models F \rightarrow G$ (F entails G iff $F \rightarrow G$ is valid)

  Property 2 (Deduction theorem)

  $N \models F$ iff $\models (F_1 \wedge \ldots \wedge F_n) \rightarrow F$

- Aside: In general Property 3

  1. $F \equiv G$ iff $\models \forall F \leftrightarrow \forall G$
  2. $F \models G$ iff $\models \forall F \rightarrow \forall G$

  Property 4 (Deduction theorem)

  $N \models F$ iff $\models (\forall F_1 \wedge \ldots \wedge \forall F_n) \rightarrow \forall F$

  $\forall F$ is the univeral closure of F; e.g., if $F = P(x)$ then $\forall F = \forall x\, P(x)$

Duality between validity and satisfiability

Recall:

- F is satisfiable, if *for some interpr.* $\mathcal{I}$ *and some* β, $\mathcal{I}_\beta(F) = \mathbf{1}$.
- F is unsatisfiable, if *for all interpr.* $\mathcal{I}$ *and all* β, $\mathcal{I}_\beta(F) = \mathbf{0}$.
- F is valid, if *for every interpretation* $\mathcal{I}$, $\mathcal{I} \models F$.

Property 5

1. F is valid iff $\neg F$ is unsatisfiable.
2. $N \models F$ iff $N \cup \{\neg F\}$ is unsatisfiable.

- Validity and unsatisfiability can be interreduced.
- Hence in order to design a theorem prover (validity checker) it is sufficient to design a checker for (un)satisfiability.

Previously ...

- syntax of first-order logic
- substitutions
  - $\{x_1/s_1, \ldots, x_n/s_n\}$
- semantics of first-order logic (standard interpretations)
- semantic entailment, semantic equivalence

- reasoning for propositional logic using resolution
- resolution calculi operate on clauses
- conversion to clausal form for propositional formulae

Normal forms

Recall: $\rightarrow$ is 'superfluous' since it can be expressed using $\neg$ and $\vee$

$$
\begin{aligned}
A \rightarrow B \;&\equiv\; \neg A \vee B, \\
&\equiv\; \neg(A \wedge \neg B), \\
&\equiv\; \neg\neg(A \rightarrow B), \\
&\text{etc}
\end{aligned}
$$

Problem: Formulas have *very many* equivalent forms

Solution: Transform formulas to normal form

- Simplifies formulas: formulas limited to fixed simple patterns
- Easier to define and determine truth
- Easier development of efficient automated reasoning tools, without penalty

Main problem in first-order logic

- ... is the treatment of quantifiers.
- Solution: Use additional normal form transformations to eliminate the quantifiers
  - Prenxe normal form transformation
  - Skolemisation

Prenex Normal Form

- A formula is in prenex normal form (PNF) if it is in the form

$$\mathcal{Q}_1 x_1 \ldots \mathcal{Q}_n x_n \; F,$$

  where F is a quantifier-free formula and $\mathcal{Q}_i \in \{\forall, \exists\}$
- $\mathcal{Q}_1 x_1 \ldots \mathcal{Q}_n x_n$ is called the quantifier prefix and F the matrix of the formula.

Useful semantic equivalences involving $\forall$, $\exists$

Let F and G be any closed formulae. Then

1. $\models \neg\forall x F \leftrightarrow \exists x \neg F$

   $\models \neg\exists x F \leftrightarrow \forall x \neg F$

2. $\models \forall x \forall y F \leftrightarrow \forall y \forall x F$

   $\models \exists x \exists y F \leftrightarrow \exists y \exists x F$

   But only: $\models \exists x \forall y F \rightarrow \forall y \exists x F$

3. $\models \forall x (F \wedge G) \leftrightarrow (\forall x F) \wedge (\forall x G)$

   $\models \exists x (F \vee G) \leftrightarrow (\exists x F) \vee (\exists x G)$

   But: $\not\models \forall x (F \vee G) \leftrightarrow (\forall x F) \vee (\forall x G)$

   $\not\models \exists x (F \wedge G) \leftrightarrow (\exists x F) \wedge (\exists x G)$

Useful semantic equivalences involving $\forall$, $\exists$ (cont'd)

4. If x does not occur freely in G then

   $\models \forall x (F \star G) \leftrightarrow (\forall x F \star G)$ for $\star \in \{\wedge, \vee\}$

   $\models \exists x (F \star G) \leftrightarrow (\exists x F \star G)$ for $\star \in \{\wedge, \vee\}$

5. If x does not occur freely in G then

   $\models \forall x (F \rightarrow G) \leftrightarrow (\exists x F \rightarrow G)$

   $\models \exists x (F \rightarrow G) \leftrightarrow (\forall x F \rightarrow G)$

   $\models \forall x (G \rightarrow F) \leftrightarrow (G \rightarrow \forall x F)$

   $\models \exists x (G \rightarrow F) \leftrightarrow (G \rightarrow \exists x F)$

Useful semantic equivalences involving $\forall$, $\exists$ (cont'd)

6. $\models \forall x F \leftrightarrow \forall y F\{x/y\}$ where y is a fresh variable

   $\models \exists x F \leftrightarrow \exists y F\{x/y\}$ where y is a fresh variable

7. $\models (\forall x F) \vee (\forall x G) \leftrightarrow \forall x \forall y (F \vee G\{x/y\})$

   where y is a fresh variable

   $\models (\exists x F) \wedge (\exists x G) \leftrightarrow \exists x \exists y (F \wedge G\{x/y\})$

   where y is a fresh variable

Computing prenex normal form

- The prenex normal form $\mathrm{PNF}(F)$ of a formula F can be computed by applying these rewrite rules:

$$
\begin{array}{rcll}
(F \leftrightarrow G) & \Rightarrow_{\mathrm{PNF}} & (F \rightarrow G) \wedge (G \rightarrow F) & \\
\neg \mathcal{Q} x F & \Rightarrow_{\mathrm{PNF}} & \overline{\mathcal{Q}} x \neg F & (\neg \mathcal{Q}) \\
(\mathcal{Q} x F \star G) & \Rightarrow_{\mathrm{PNF}} & \mathcal{Q} y (F\{x/y\} \star G), \quad \star \in \{\wedge, \vee\} & \\
(F \star \mathcal{Q} x G) & \Rightarrow_{\mathrm{PNF}} & \mathcal{Q} y (F \star G\{x/y\}), \quad \star \in \{\wedge, \vee, \rightarrow\} & \\
(\mathcal{Q} x F \rightarrow G) & \Rightarrow_{\mathrm{PNF}} & \overline{\mathcal{Q}} y (F\{x/y\} \rightarrow G), &
\end{array}
$$

  In the last three lines y must be a fresh variable in each case.
- $\overline{\mathcal{Q}}$ denotes the quantifier dual to $\mathcal{Q}$, i.e., $\overline{\forall} = \exists$ and $\overline{\exists} = \forall$.
- The rules can be applied in any order

Exercise

- Obtain the prenex normal form for the formula

$$\exists x \forall y \big(\exists z (P(x,z) \wedge P(y,z)) \rightarrow \exists u Q(x,y,u)\big).$$

$$\exists x \forall y \big(\exists z (P(x,z) \wedge P(y,z)) \rightarrow \exists u Q(x,y,u)\big)$$

$$\Rightarrow_{\mathrm{PNF}}$$

Eliminating $\exists$ quantifiers using Skolemisation

- Transformation $\Rightarrow_{\mathrm{Sk}}$

$$\forall x_1 \ldots \forall x_n \exists y F \quad \Rightarrow_{\mathrm{Sk}} \quad \forall x_1 \ldots \forall x_n F\{y/f(x_1,\ldots,x_n)\}$$

  where f/n is a fresh function symbol.

- f is called a Skolem function.
  $f(x_1,\ldots,x_n)$ is called a Skolem term, or Skolem constant when $n=0$.
- Example:

$$\forall x \exists y P(x,y) \;\; \Rightarrow_{\mathrm{Sk}} \;\; \forall x P(x, f(x))$$

- Intuition: f is choice function computing y from all the arguments that y depends on.
- **Note:** Always apply outermost first, not in subformulae

Exercise

- Apply Skolemisation to this formula

$$\exists x \forall y \forall z \exists u \forall v \; P(x, y, z, u, v)$$

Applying transformation to PNF and Skolemising

Together: $F \Rightarrow^*_{\text{PNF}} \underbrace{G}_{\text{PNF}} \Rightarrow^*_{\text{Sk}} \underbrace{H}_{\text{PNF , no } \exists}$

Property 6

Let F, G, and H be as above and closed. Then

(i) F and G are equivalent.

(ii) $H \models G$, but the converse is not true in general.

(iii) G is satisfiable iff H is satisfiable

In the last case: In fact, G is satisfiable in an interpretation $\mathcal{I}$ iff H is satisfiable in an extenstion $\mathcal{I}'$ of $\mathcal{I}$.
$\mathcal{I}$ is an interpr. over $\Sigma = (\mathcal{F}, \mathcal{P})$, while
$\mathcal{I}'$ is an interpr. over $\Sigma' = (\mathcal{F} \cup SKF, \mathcal{P})$.

Literals, clauses

- Literals

$$\begin{array}{rcll} L & \longrightarrow & A & \text{(atom, positive literal)} \\ & | & \neg A & \text{(negative literal)} \end{array}$$

- Clauses

$$\begin{array}{rcll} C, D & \longrightarrow & \bot & \text{(empty clause)} \\ & | & L_1 \vee \ldots \vee L_k, \;\; k \geq 1 & \text{(non-empty clause)} \end{array}$$

- **Important assumptions:**
  - $\vee$ is associative and commutative, repetitions matter.
    I.e. we regard clauses as multi-sets of literals
  - Thus, $C = P \vee P \vee \neg Q$ is identical to $C' = P \vee \neg Q \vee P$.
    But neither C nor C' are the same as $D = P \vee \neg Q$.

Transformation to conjunctive normal form

- The conjunctive normal form CNF(F) of a formula F can be computed by persistently applying these rewrite rules:

$$\begin{array}{rcllrcl} F \leftrightarrow G & \Rightarrow_{\text{CNF}} & (F \rightarrow G) \wedge (G \rightarrow F) & & & & \\ F \rightarrow G & \Rightarrow_{\text{CNF}} & (\neg F \vee G) & & & & \\ \neg(F \vee G) & \Rightarrow_{\text{CNF}} & (\neg F \wedge \neg G) & & & & \\ \neg(F \wedge G) & \Rightarrow_{\text{CNF}} & (\neg F \vee \neg G) & & & & \\ \neg\neg F & \Rightarrow_{\text{CNF}} & F & & & & \\ (F \wedge G) \vee H & \Rightarrow_{\text{CNF}} & (F \vee H) \wedge (G \vee H) & & & & \\ F \wedge \top & \Rightarrow_{\text{CNF}} & F & & F \wedge \bot & \Rightarrow_{\text{CNF}} & \bot \\ F \vee \top & \Rightarrow_{\text{CNF}} & \top & & F \vee \bot & \Rightarrow_{\text{CNF}} & F \\ \neg\top & \Rightarrow_{\text{CNF}} & \bot & & \neg\bot & \Rightarrow_{\text{CNF}} & \top \end{array}$$

- These rules are to be applied modulo associativity and commutativity of $\wedge$ and $\vee$.

Computing the clausal form of a first-order formula

$$
\begin{aligned}
F \;&\Rightarrow^*_{\mathrm{PNF}}\; \mathcal{Q}_1 y_1 \ldots \mathcal{Q}_n y_n\; G && (G \text{ quantifier-free})\\
&\Rightarrow^*_{\mathrm{Sk}}\; \forall x_1 \ldots \forall x_m\; H && (m \le n,\ H \text{ quantifier-free})\\
&\Rightarrow^*_{\mathrm{CNF}}\; \underbrace{\underbrace{\forall x_1 \ldots \forall x_m}_{\text{leave out}} \; \bigwedge_{i=1}^{k} \; \underbrace{\bigvee_{j=1}^{n_i} L_{ij}}_{\text{clause } C_i}}_{F'}\\
&\Rightarrow\; \{C_1, \ldots, C_k\}
\end{aligned}
$$

- $N = \{C_1, \ldots, C_k\}$ is called the clausal (normal) form of F.
- **Note:** the variables in the clauses are implicitly universally quantified.

Sample transformation to clausal form

- Given formula:
  $$\exists x \big[\forall y\, (R(x,y) \rightarrow (\neg P(y) \vee \exists z\, (R(y,z) \wedge P(z))))\big]$$
- Prenex normal form:
  $$\exists x \forall y \exists z\, \big[R(x,y) \rightarrow (\neg P(y) \vee (R(y,z) \wedge P(z)))\big]$$
- Skolemisation:
  $$\forall y\, \big[R(a,y) \rightarrow (\neg P(y) \vee (R(y,f(y)) \wedge P(f(y))))\big]$$
  Sk. const. for $\exists x$ Sk. term for $\exists z$
- CNF:
  $$\forall y\, \big[(\neg R(a,y) \vee \neg P(y) \vee R(y,f(y))) \wedge (\neg R(a,y) \vee \neg P(y) \vee P(f(y)))\big]$$
- Clausal form: drop $\forall$, $\wedge$ and outer brackets
  $$\neg R(a,y) \vee \neg P(y) \vee R(y,f(y))$$
  $$\neg R(a,y) \vee \neg P(y) \vee P(f(y))$$

Properties of CNFs and clausal forms

Property 7

For every formula F:

$$\text{If} \quad F \Rightarrow^*_{\mathrm{CNF}} F' \quad \text{then} \quad F \equiv F'.$$

Property 8

Let F be closed. Suppose $F \Rightarrow^*_{\mathrm{PNF}} \circ \Rightarrow^*_{\mathrm{Sk}} \circ \Rightarrow^*_{\mathrm{CNF}} F'$ and N is clausification of F'.

$$\text{Then} \quad F' \models F \quad \text{and} \quad N \models F.$$

The converses are not true in general. But:

Property 9

Let F be closed. Then

F is satisfiable iff F' is satisfiable iff N is satisfiable

Optimising the transformation to clausal form

- Issues:
  - Size of the CNF can be exponential;
  - Want to preserve the original formula structure;
  - Want Skolem functions with small arity.
- These can all be addessed since we can/need to preserve only satisfiability anyway $\rightsquigarrow$ lots of room for optimisation
- The last point can be addressed with miniscoping of quantifiers (essentially moving quantifiers inwards) and a better form of $\Rightarrow_{\mathrm{Sk}}$ (not discussed)
- The first two points can be addressed with structural transformation (idea similar as for propositional logic; not discussed)

Previously ...

- Every FO formula F can be transformed into a set N of clauses so that

$$F \text{ is satisfiable} \quad \text{iff} \quad N \text{ is satisfiable.}$$

- We will extend resolution to first-order clauses

$$\neg R(a, y) \vee \neg P(y) \vee R(y, f(y))$$
$$\neg R(a, y) \vee \neg P(y) \vee P(f(y))$$

Herbrand semantics for first-order clauses

- Problem with classical semantics: There are soooo many ways to define interpretations
- Herbrand interpretations are a special interpretations that allow for a very simple definition and analysis of the truth of clauses.

- Key idea of Herbrand's theorem:
  It suffices that terms are interpreted as themselves.
  For establishing the truth of clauses it suffices to consider only Herbrand interpretations.

Ground expressions, ground instances

- Ground terms are terms with no occurrences of variables.
- Ground atoms are atoms with no occurrences of variables.
- Ground literals, ground clauses, ground formulae are defined similarly.
- A ground instance of an expression (term, atom, literal, clause, formula) is obtained by uniformly substituting the variables in it with ground terms.

Herbrand universe

- Herbrand semantics allows us to fix a special domain s.t. if F is unsatisfiable, then every truth assignment over this special domain is false.
- The Herbrand universe is T_Σ, i.e., the set of all ground terms over the signature $\Sigma = (\mathcal{F}, \mathcal{P})$.
- Example: Suppose $\mathcal{F}$ has one binary function symbol f and two constants a and b. Herbrand universe over Σ:

$$a,\ b,\ f(a,a),\ f(a,b),\ f(b,a),\ f(b,b),\ f(a,f(a,a)),\ \ldots$$

- If Σ contains non-constant function symbols then T_Σ is infinite.
- Important assumption: There is at least one constant in the signature Σ.

Exercise

- Suppose Σ is a signature with one unary function symbol f and one constant a. I.e. $\mathcal{F} = \{f/1, a/0\}$.
  Write down the elements of the Herbrand universe T_Σ.

Herbrand interpretations

- A Herbrand interpretation, denoted I, is a set of ground atoms over Σ.
- Truth in I of ground formulae is defined inductively by:

$$I \models \top \qquad\qquad I \not\models \bot$$

$$(*) \qquad I \models A \text{ iff } A \in I\text{, for any ground atom } A$$

$$I \models \neg F \text{ iff } I \not\models F$$

$$I \models F \land G \text{ iff } I \models F \text{ and } I \models G$$

$$I \models F \lor G \text{ iff } I \models F \text{ or } I \models G$$

- Note: $(*)$ is equivalent to $I \not\models A$ iff $A \notin I$
- This means: $A \notin I$ implies A is false in I, which implies $I \models \neg A$

Herbrand interpretations (cont'd)

- Truth in I of any quantifier-free formula F with free variables $x_1, \dots, x_n$ is defined by:

$$I \models F(x_1, \dots, x_n) \text{ iff } I \models F(t_1, \dots, t_n), \text{ for every } t_i \in \mathrm{T}_\Sigma$$

- Truth in I of any set N of clauses/quantifier-free formulae is defined by:

$$I \models N \quad \text{iff} \quad I \models C, \text{ for each } C \in N$$

- A Herbrand interpretation I is called a Herbrand model of F, if $I \models F$.

Exercise

- Suppose $\mathcal{F} = \{f/1, a/0\}$ and $\mathcal{P} = \{P/1\}$.
  Which of the following are Herbrand interpretations over Σ?
  1. $I_1 = \{P(a)\}$
  2. $I_2 = \{P(a),\ P(f(a))\}$
  3. $I_3 = \{P(a),\ \neg P(f(a))\}$
- For I_2 determine whether the following is true?
  1. $I_2 \models P(a)$
  2. $I_2 \models \neg P(a)$
  3. $I_2 \models \neg P(f(a))$
  4. $I_2 \models P(a) \wedge P(f(a))$
  5. $I_2 \models P(x)$

Examples of truth in Herbrand interpretations

Suppose Σ is any signature. Let I be a Herbrand interpretation over Σ.

- $I \models P(x)$ iff $P(t) \in I$ for every ground term $t \in \mathrm{T}_\Sigma$.
- $I \models P(x) \vee Q(x)$ iff for every ground term $t \in \mathrm{T}_\Sigma$ $P(t) \in I$ or $Q(t) \in I$.
- $I \models \neg S(x, a)$ iff $S(t, a) \notin I$ for every ground term t in T_Σ.
- $I \models \neg R(x, y) \vee R(y, x)$ iff if $R(s, t) \in I$ then $R(t, s) \in I$, for any ground terms $s, t \in \mathrm{T}_\Sigma$.

Relation to standard interpretations (aside)

- In a Herbrand interpretation values are fixed to be ground terms and functions are fixed to be the (Skolem) functions in Σ.
- Let I be a Herbrand interpretation over Σ with domain T_Σ.
- Let $\mathcal{I} = (U, \cdot^{\mathcal{I}})$ be a standard interpretation where $U = \mathrm{T}_\Sigma$ and the interpretation function $\cdot^{\mathcal{I}}$ maps terms to themselves, i.e.:

  constant a: $a^{\mathcal{I}} = a$

  function f: $f^{\mathcal{I}}(t_1, \ldots, t_n) = f(t_1, \ldots, t_n)$

  Only predicate symbols P may be freely interpreted as relations $P^{\mathcal{I}}$ over T_Σ.

Property 10

Every Herbrand interpretation I (set of ground atoms) uniquely determines a standard interpretation $\mathcal{I}$ via

$$P^{\mathcal{I}}(s_1, \ldots, s_n) = \mathbf{1} \quad \text{iff} \quad P(s_1, \ldots, s_n) \in I$$

The set of all ground instances $G_\Sigma(N)$

- Let N be a set clauses over the signature Σ and suppose $\mathcal{X}$ denotes the set of variables in N. Define

  $$G_\Sigma(N) = \{C\sigma \mid C \in N, \sigma : \mathcal{X} \to \mathrm{T}_\Sigma \text{ a ground substitution}\}$$

  $G_\Sigma(N)$ is the set of all ground instances of the clauses in N.

- Example: $N = \{P(x), \; Q(f(y)) \vee R(y)\}$

  $T_\Sigma = \{a, f(a), f(f(a)), \ldots\}$

  - $P(a)$ and $P(f(a))$ are both ground instances of the first clause $P(x)$ in N.
  - **Exercise**: Give examples of ground instances of the second clause in N.

Existence of Herbrand models

Property 11 (Herbrand)

Let N be a set of Σ-clauses. Then

N is true in a standard interpretation

iff N has a Herbrand model (over Σ)

iff $G_\Sigma(N)$ has a Herbrand model (over Σ)

- Many theorem proving approaches exploit this property, e.g., approaches based on instantiation, e.g., tableau approaches, Inst-Gen, but also resolution.
- We use it in the completeness proof of the resolution calculus.

Using Herbrand's theorem to find a model

- Suppose the signature is based on $\mathcal{F} = \{1/0, +/2\}$ and $\mathcal{P} = \{P/1\}$. Is the following set satisfiable?

$$N = \{\ P(1),\ \neg P(x) \vee P(x+1)\ \}$$

- One obtains the following ground instances:

$$\begin{aligned} G_\Sigma(N) = \{\ & P(1), \\ & \neg P(1) \vee P(1+1), \\ & \neg P(1+1) \vee P(1+1+1), \\ & \neg P(1+1+1) \vee P(1+1+1+1), \\ & \ldots \\ & \} \end{aligned}$$

Exercise

- Write down a Herbrand model of $G_\Sigma(N)$.
  I.e. write down a Herbrand interpretation in which all clauses of $G_\Sigma(N)$ are true.

- Is N satisfiable?

Summary

- ground expressions, ground instances
- Herbrand universe
- Herbrand interpretation, Herbrand model
- truth in I: $\models$
- Herbrand's theorem

Literals, clauses

- Literals

$$\begin{array}{rcll} L & \longrightarrow & A & \text{(atom, positive literal)} \\ & | & \neg A & \text{(negative literal)} \end{array}$$

- Clauses

$$\begin{array}{rcll} C, D & \longrightarrow & \bot & \text{(empty clause)} \\ & | & L_1 \vee \ldots \vee L_k,\ \ k \geq 1 & \text{(non-empty clause)} \end{array}$$

- **Important assumptions:**
  - $\vee$ is associative and commutative, repetitions matter.
    I.e. we regard clauses as multi-sets of literals
  - Thus, $C = P \vee P \vee \neg Q$ is identical to $C' = P \vee \neg Q \vee P$.
    But neither C nor C' are the same as $D = P \vee \neg Q$.

The unrestricted resolution calculus Res

- **Propositional/ground resolution calculus** *Res*

$$\frac{C \vee A \qquad \neg A \vee D}{C \vee D} \qquad \text{(resolution)}$$

$$\frac{C \vee A \vee A}{C \vee A} \qquad \text{((positive !) factoring)}$$

- Terminology: $C \vee D$ is the resolvent
  $C \vee A$ is the (positive) factor
  A is the atom resolved upon, resp. factored upon
- Since we assume $\vee$ is associative and commutative, note that A and $\neg A$ can occur anywhere in their respective clauses.

Recall: Soundness and completeness

- Our aim: prove that *Res* is sound and refutationally complete.
- $N \models C$ means every model of N is also a model of C
  C is true in each model of N; C follows semantically from N
  $N \models \bot$ means N is unsatisfiable, i.e. N has no model.
  $N \vdash_{Cal} C$ means there is a finite *Cal*-derivation of C from N.
  $N \vdash_{Res} C$ means there is a finite *Res*-derivation of C from N.
- *Cal* is said to be sound iff

$$N \vdash_{Cal} C \text{ implies } N \models C.$$

- *Cal* is complete iff $N \models C$ implies $N \vdash_{Cal} C$.
- *Cal* is said to be refutationally complete iff

$$N \models \bot \text{ implies } N \vdash_{Cal} \bot.$$

Sound inference rule

- An inference rule

$$\frac{F_1 \ \ldots \ F_n}{F}$$

is called sound, if $F_1, \ldots, F_n \models F$,
i.e., F is a semantic/logical consequence of $F_1 \wedge \ldots \wedge F_n$.

Soundness of resolution

Property 12

The propositional resolution calculus *Res* (resolution on ground clauses) is sound.

Proof: We have to show: $N \vdash_{Res} C$ implies $N \models C$.
It suffices to show that every rule is sound, i.e. for every rule $\frac{C_1 \ \ldots \ C_n}{D}$ we have $C_1, \ldots, C_n \models D$.
For resolution, assume $I \models C \vee A$, $I \models \neg A \vee D$ and show $I \models C \vee D$.
(a) Case $I \models A$: Then $I \models D$, for else $I \not\models \neg A \vee D$. Hence $I \models C \vee D$. (b) Case $I \not\models A$: Since $I \models C \vee A$, $I \models C$ and consequently $I \models C \vee D$.
For factoring, assume $I \models C \vee A \vee A$ and show $I \models C \vee A$. Exercise.

Refutational completeness of resolution

- How to show refutational completeness of ground resolution?
- We have to show: $N \models \bot$ implies $N \vdash_{Res} \bot$,
  or equivalently: $N \not\vdash_{Res} \bot$ implies N has a model.
- **Idea:**
  - Suppose that we have computed all possible inferences from N (and not derived $\bot$); could be infinitely many inferences.
  - Order the clauses in the derivation according to an appropriate ordering, inspect the clauses in ascending order, and construct a series of Herbrand interpretations.
  - The limit Herbrand interpretation can be shown to be a model of N.

Defining ground literal and clause orderings

- We assume that $\succ$ is any fixed ordering on ground atoms that is total and well-founded. (There exist many such orderings, e.g., the length-based ordering on atoms when these are viewed as words over a suitable alphabet.)
- Extend $\succ$ to an ordering $\succ_L$ on ground literals:

$$\begin{array}{lcll} [\neg]A & \succ_L & [\neg]B, & \text{if } A \succ B \\ \neg A & \succ_L & A & \end{array}$$

  (These are 5 conditions!)
- Extend $\succ_L$ to an ordering $\succ_C$ on ground clauses:
  Let $\succ_C = (\succ_L)_{\mathrm{mul}}$, the multi-set extension of $\succ_L$.
- Notation: $\succ$ also for $\succ_L$ and $\succ_C$.

Recap: Multi-set extension ordering of an ordering

- Let $(X, \succ)$ be an ordering. The multi-set extension $\succ_{\mathrm{mul}}$ of $\succ$ to (finite) multi-sets over X is defined by

$$\begin{array}{lll} S_1 \succ_{\mathrm{mul}} S_2 & \text{iff} & S_1 \neq S_2 \text{ and} \\ & & \forall x \in S_2 \backslash S_1.\ \exists y \in S_1 \backslash S_2.\ y \succ x \end{array}$$

- Cancellation method for determining $S_1 \succ_{\mathrm{mul}} S_2$:
  1. Remove common occurrences of elements from S_1 and S_2. Assume this gives S_1' and S_2'.
  2. Then check that for every element x in S_2' there is an element $y \in S_1'$ that is larger than x. Then $S_1 \succ_{\mathrm{mul}} S_2$.

Example

- Suppose $A_5 \succ A_4 \succ A_3 \succ A_2 \succ A_1 \succ A_0$.
- Then:

$$\neg A_5 \succ A_5 \succ \neg A_4 \succ A_4 \succ \ldots \succ \neg A_0 \succ A_0$$

- And:

$$\begin{array}{cc} & A_0 \vee A_1 \\ \prec & A_1 \vee A_2 \\ \prec & \neg A_1 \vee A_2 \\ \prec & \neg A_1 \vee A_4 \vee A_3 \\ \prec & \neg A_1 \vee \neg A_4 \vee A_3 \\ \prec & \neg A_5 \vee A_5 \end{array}$$

Exercise

- Suppose $A_4 \succ A_3 \succ A_2 \succ A_1$
- How are these clauses ordered by $\succ_C$?

1. $\neg A_3 \vee A_4$
2. $A_3 \vee A_1 \vee A_1$
3. $\neg A_4 \vee A_2$
4. $A_3 \vee A_1$

Properties of clause orderings

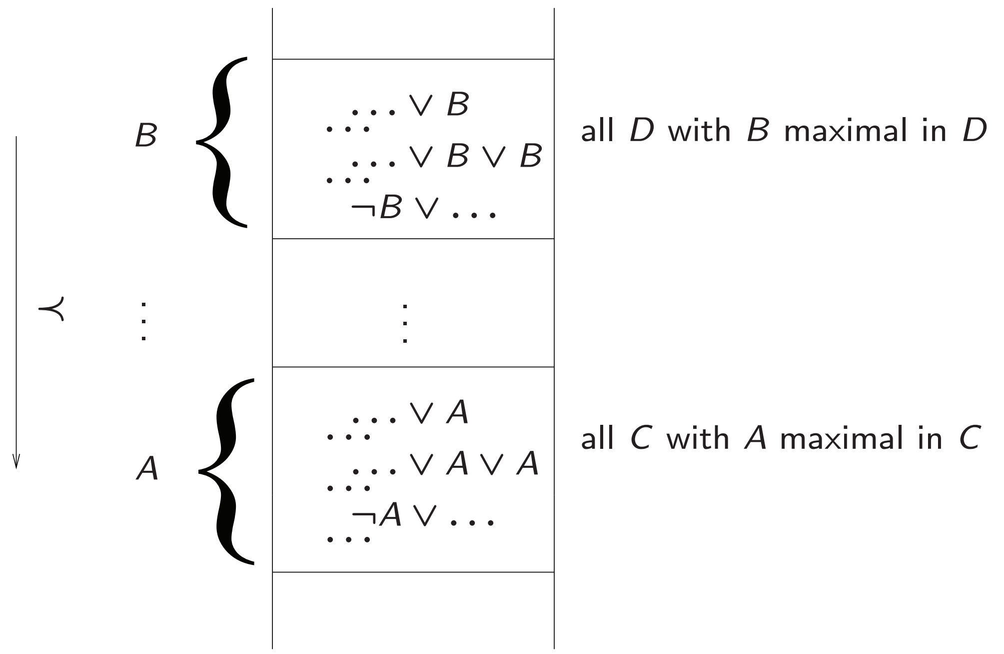
Property 13

1. The orderings ($\succ_L$ and $\succ_C$) on ground literals and clauses are total and well-founded.
2. Let C and D be clauses with A an occurrence of a maximal atom in C and B an occurrence of a maximal atom in D.
   (i) If $A \succ B$ then $C \succ D$.
   (ii) If $A = B$ and A occurs negatively in C but only positively in D, then $C \succ D$.

Note: in 2. A and B may be negated or unnegated occurrences.

Stratified structure of clause sets

Let $A \succ B$. Clause sets are then stratified in this form:



Clauses in A-cluster are larger than clauses in B-cluster

Method of level saturation

- $\ldots =$ a method of computing all possible conclusions
- Let $\quad Res(N) = \{C \mid C$ is the conclusion of applying a rule in Res to premises in $N\}$
  $Res(N)$ is the set of 'immediate' resolvents and factors of N (all premises are in N).
- Define N_n and Res^* by:

$$
\begin{aligned}
N_0 &= N \\
N_{n+1} &= N_n \cup Res(N_n), \quad \text{for } n \geq 0 \\
Res^*(N) &= \bigcup_{n \geq 0} N_n
\end{aligned}
$$

  $Res^*(N)$ is the set of all possible resolvents and factors of N.

Saturation of clause sets under *Res*

- N is called saturated (wrt. Res), if

$$Res(N) \subseteq N.$$

- The method of level saturation computes the saturation of a set N as the deductive closure of N which is given by $Res^*(N)$.

Property 14

(i) $Res^*(N)$ is saturated.

(ii) Res is sound and refutationally complete iff for each set N of ground clauses:

$$N \models \perp \ \text{ iff } \ \perp \in Res^*(N)$$

- Intuition: $\perp \in Res^*(N)$ implies $N \vdash_{Res} \perp$

Previously ...

- soundness and refutational completeness
- sound rule
- soundness of *Res*
- literal ordering, clause ordering
- properties of ordered clause sets, stratification
- saturated clause set, level saturation

Construction of Herbrand interpretations

- Our aim is to show the equivalence, where N is any set of ground clauses:

$$N \models \bot \;\text{ iff }\; \bot \in \mathit{Res}^*(N)$$

- The soundness result (Property 12) implies the "$\Leftarrow$" direction.
- We now show the "$\Rightarrow$" direction (i.e. refutational completeness), by showing

  If $\bot \notin \mathit{Res}^*(N)$, then N has a model.

- **Given**: set N of ground clauses, atom ordering $\succ$.
- **Wanted**: Herbrand interpretation I such that
  - "many" clauses from N are true in I, and
  - $I \models N$, if N is saturated and $\bot \notin N$.

Example

Let $A_5 \succ A_4 \succ A_3 \succ A_2 \succ A_1 \succ A_0$ (strictly maximal literals in *red*)

	clauses C in N	I_C	Δ_C	Remarks
1	$\neg A_0$	$\emptyset$	$\emptyset$	true in I_C
2	$A_0 \vee A_1$	$\emptyset$	$\{A_1\}$	A_1 str. maximal
3	$A_1 \vee A_2$	$\{A_1\}$	$\emptyset$	true in I_C
4	$\neg A_1 \vee A_2$	$\{A_1\}$	$\{A_2\}$	A_2 str. maximal
5	$\neg A_1 \vee A_4 \vee A_3 \vee A_0$	$\{A_1, A_2\}$	$\{A_4\}$	A_4 str. maximal
6	$\neg A_1 \vee \neg A_4 \vee A_3$	$\{A_1, A_2, A_4\}$	$\emptyset$	A_3 not str. max. *min. exception*
7	$\neg A_1 \vee A_5$	$\{A_1, A_2, A_4\}$	$\{A_5\}$	A_5 str. maximal

$I = \{A_1, A_2, A_4, A_5\}$ is not a model of the clause set
because there exists an exception (unfulfilled) clause, clause 6.

By definition, an exception clause for I is a clause that is not true in I.

Main ideas of the construction

- Approximate (!) description: Define I inductively by:
  - Starting with a minimal clause C in N.
    (Since in the ground case the ordering is total, there is a smallest clause and we start in fact with this clause.)
  - Consider the largest atom in C and attempt to define (in a certain way) $I_C \cup \Delta_C$ (!) as the minimal extension of the partial interpretation constructed so far (I_C) so that C becomes true.
  - Iterate for $N \backslash \{C\}$, and so forth.
- I.e. clauses are considered in the order given by $\prec$.
- When considering C, one already has a partial interpretation I_C available (initially $I_C = \emptyset$).

Main ideas of the construction (cont'd)

- If C is true in the partial interpretation I_C, nothing is done ($\Delta_C = \emptyset$).
- If C is false, change I_C such that C becomes true.
- Changes should, however, be monotone. One never deletes anything from I_C and the truth value of any clause smaller than C should be maintained the way it was in I_C.
- Hence, one chooses $\Delta_C = \{A\}$ iff C is false in I_C, and when both
  - (i) A occurs positively in C, and
  - (ii) this occurrence of A in C is strictly maximal (i.e. largest) in the ordering on literals.
- Note: (i) implies adding A will make C become true.
  (ii) implies changing the truth value of A has no effect on smaller clauses.

Resolution reduces exceptions

$$\frac{\neg A_1 \lor \underline{A_4} \lor A_3 \lor A_0 \qquad \neg A_1 \lor \underline{\neg A_4} \lor A_3}{\neg A_1 \lor \neg A_1 \lor A_3 \lor A_3 \lor A_0}$$

Construction of I for the extended clause set:

clauses C	I_C	Δ_C	Remarks
$\underline{\neg A_0}$	$\emptyset$	$\emptyset$	
$A_0 \lor \underline{A_1}$	$\emptyset$	$\{A_1\}$	
$A_1 \lor \underline{A_2}$	$\{A_1\}$	$\emptyset$	
$\neg A_1 \lor \underline{A_2}$	$\{A_1\}$	$\{A_2\}$	
$\neg A_1 \lor \neg A_1 \lor \underline{A_3} \lor \underline{A_3} \lor A_0$	$\{A_1, A_2\}$	$\emptyset$	A_3 occurs twice *min. exception*
$\neg A_1 \lor \underline{A_4} \lor A_3 \lor A_0$	$\{A_1, A_2\}$	$\{A_4\}$	
$\neg A_1 \lor \underline{\neg A_4} \lor A_3$	$\{A_1, A_2, A_4\}$	$\emptyset$	exception
$\neg A_1 \lor \underline{A_5}$	$\{A_1, A_2, A_4\}$	$\{A_5\}$	

The same I, but smaller exception, hence some progress was made.

Factoring reduces exceptions

$$\frac{\neg A_1 \vee \neg A_1 \vee \underline{A_3} \vee \underline{A_3} \vee A_0}{\neg A_1 \vee \neg A_1 \vee \underline{A_3} \vee A_0}$$

Construction of I for the extended clause set:

clauses C	I_C	Δ_C	Remarks
$\underline{\neg A_0}$	$\emptyset$	$\emptyset$	
$A_0 \vee \underline{A_1}$	$\emptyset$	$\{A_1\}$	
$A_1 \vee \underline{A_2}$	$\{A_1\}$	$\emptyset$	
$\neg A_1 \vee \underline{A_2}$	$\{A_1\}$	$\{A_2\}$	
$\neg A_1 \vee \neg A_1 \vee \underline{A_3} \vee A_0$	$\{A_1, A_2\}$	$\{A_3\}$	
$\neg A_1 \vee \neg A_1 \vee \underline{A_3} \vee \underline{A_3} \vee A_0$	$\{A_1, A_2, A_3\}$	$\emptyset$	true in I_C
$\neg A_1 \vee \underline{A_4} \vee A_3 \vee A_0$	$\{A_1, A_2, A_3\}$	$\emptyset$	
$\neg A_1 \vee \underline{\neg A_4} \vee A_3$	$\{A_1, A_2, A_3\}$	$\emptyset$	true in I_C
$\neg A_3 \vee \underline{A_5}$	$\{A_1, A_2, A_3\}$	$\{A_5\}$	

The resulting $I = \{A_1, A_2, A_3, A_5\}$ is a model of the clause set.

Construction of candidate models formally

- Let $N, \succ$ be given. Guided by $\succ$, we define sets I_C and Δ_C for all ground clauses C over the given signature inductively by:

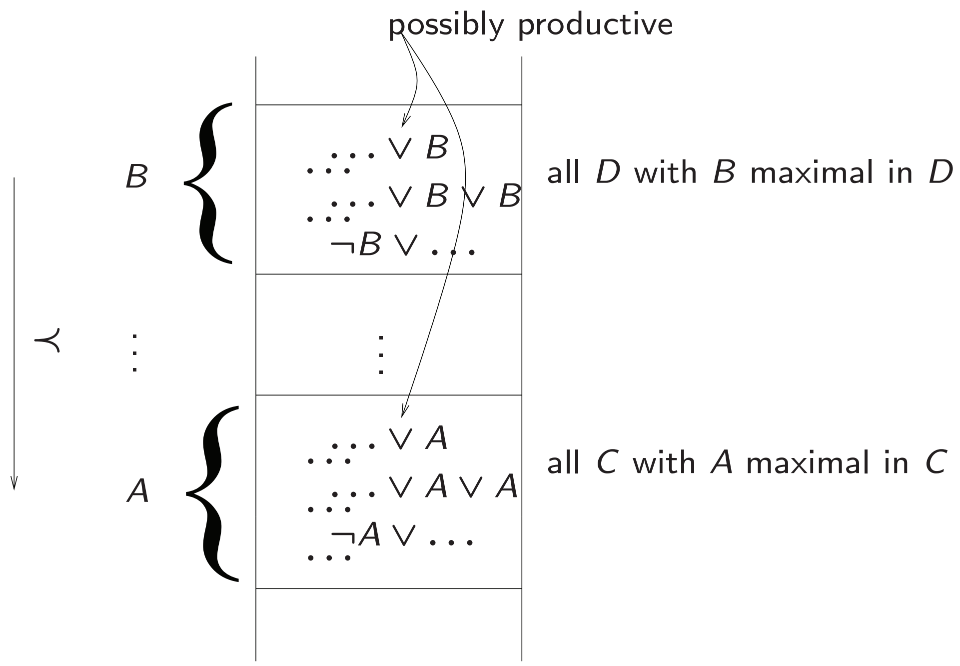
$$I_C := \bigcup_{C \succ D} \Delta_D$$

$$\Delta_C := \begin{cases} \{A\}, & \text{if } C \in N,\ C = C' \vee A, \\ & A \succ C' \text{ and } I_C \not\models C \\ \emptyset, & \text{otherwise} \end{cases}$$

- We say, C produces A, or just C is productive, if $\Delta_C = \{A\}$.
- The candidate model for N (wrt. $\succ$) is given as $I_N^{\succ} := \bigcup_{C \in N} \Delta_C$.
- We also simply write I_N, or I, for $I_N^{\succ}$, if $\succ$ is either irrelevant or known from the context.

Structure of $(N, \succ)$

Let $A \succ B$; producing a new atom does not affect smaller clauses.



The smallest clauses in each cluster are possibly productive; but not necessarily (particularly if they are already true in I_C).

Some properties of the construction

Property 15

(i) $C = \neg A \vee C'$ implies no D s.t. $D \succeq C$ produces A.

(ii) C productive implies $I_C \cup \Delta_C \models C$ and $I_N \models C$.

(iii) Let $D' \succ D \succeq C$. Then

$$I_D \cup \Delta_D \models C \quad \text{implies} \quad I_{D'} \cup \Delta_{D'} \models C \text{ and } I_N \models C.$$

If, in addition, $C \in N$ or $B \succ A$, where B and A are maximal atoms in D and C, respectively, then

$$I_D \cup \Delta_D \not\models C \quad \text{implies} \quad I_{D'} \cup \Delta_{D'} \not\models C \text{ and } I_N \not\models C.$$

Some properties of the construction (cont'd)

(iv) Let $D' \succ D \succ C$. Then

$$I_D \models C \quad \text{implies} \quad I_{D'} \models C \text{ and } I_N \models C.$$

If, in addition, $C \in N$ or $B \succ A$, where B and A are maximal atoms in D and C, respectively, then

$$I_D \not\models C \quad \text{implies} \quad I_{D'} \not\models C \text{ and } I_N \not\models C.$$

(v) $C = C' \vee A$ produces A implies $I_N \not\models C'$.

Model existence theorem

Property 16 (Bachmair, Ganzinger 1990)

Let $\succ$ be a clause ordering, let N be saturated wrt. *Res*, and suppose that $\bot \notin N$. Then

$$I_N^{\succ} \models N.$$

Corollary 17

Let N be saturated wrt. *Res*. Then

$$N \models \bot \quad \text{iff} \quad \bot \in N.$$

Corollary 18

Res is refutationally complete.

Model existence theorem (cont'd)

Proof of Property 16:

Suppose $\bot \notin N$, but $I_N \not\models N$. (NB: $I_N = I_N^{\succ}$)

Let $C \in N$ be minimal (wrt. $\succ$) such that $I_N \not\models C$.

Since C is false in I_N, C is not productive.

As $C \neq \bot$, there exists a maximal atom A in C.

Case 1: $C = \neg A \vee C'$ (i.e., the maximal atom occurs negatively)

$\Rightarrow\ I_N \not\models \neg A$ and $I_N \not\models C' \ \Rightarrow\ I_N \models A$ and $I_N \not\models C'$

$\Rightarrow$ some $D = D' \vee A \in N$ produces A. As $\frac{D' \vee A \qquad \neg A \vee C'}{D' \vee C'}$, we infer that $D' \vee C' \in N$, and $C \succ D' \vee C'$ and $I_N \not\models D' \vee C'$

$\Rightarrow$ contradicts minimality of C.

Case 2: $C = C' \vee A \vee A$. Then $I_N \not\models A$ and $I_N \not\models C'$. Then $\frac{C' \vee A \vee A}{C' \vee A}$ yields a smaller exception $C' \vee A \in N$.

$\Rightarrow$ contradicts minimality of C.

Summary

- refutational completeness of *Res*
- model construction
  - given: set N of ground clauses; atom ordering $\succ$
  - output: candidate model $I_N^{\succ}$
- model existence theorem

- productive clause
- exceptions, minimal exceptions

Ground resolution based on Herbrand's theorem

- Recall Herbrand's theorem: For N any set of Σ-clauses, N is satisfiable iff $G_\Sigma(N)$ has a Herbrand model
- This means: N is (un)satisfiable iff $G_\Sigma(N)$ is (un)satisfiable
- This suggests the following semi-decision procedure using the ground resolution calculus *Res*:

```
Input: An enumeration of G_Σ(N) = {C_1, C_2, ...};
i := 0;  M := ∅;
while ⊥ ∉ M
      i := i + 1;
      M := Res*(M ∪ {C_i});
Output: unsatisfiable if ⊥ ∈ M
```

Completeness and example

Property 19

Let N be a set of general clauses. The ground resolution procedure, with $G_\Sigma(N)$ as input, terminates after a finite number of steps iff N is unsatisfiable.

N
$i.\ \neg Q(a) \vee \neg P(a)$
$ii.\ \ P(x)$
$iii.\ \neg P(f(y)) \vee Q(y)$

$G_\Sigma(N)$
$\neg Q(a) \vee \neg P(a)$,
$P(a),\ P(f(a))$,
$\neg P(f(a)) \vee Q(a)$,
$P(f(f(a))), \neg P(f(f(a))) \vee \ldots,$

Ground resolution derivation	
1. $\neg Q(a) \vee \neg P(a)$	i
2. $P(a)$	$ii\{x/a\}$
3. $\neg Q(a)$	$(1, 2)$
4. $P(f(a))$	$ii\{x/f(a)\}$
5. $\neg P(f(a)) \vee Q(a)$	$iii\{y/a\}$
6. $Q(a)$	$(4, 5)$
7. $\perp$	$(3, 6)$

Ground resolution for FO clause logic

- Ground (propositional) resolution:
  - idea: find appropriate instances of given clauses based on Herbrand's theorem
  - gives a semi-decision procedure for FOL
  - in its most naive version, is not guaranteed to terminate for satisfiable sets of clauses with equality (improved versions do terminate, however)
  - is inferior to the DPLL procedure (even with various improvements).
- But: in contrast to the DPLL procedure, resolution can be easily extended to non-ground clauses.

Issues and refined idea

- More than one instance of a clause can participate in a proof.
- Even worse: There are infinitely many possible instances.
- Observation: Instantiation must produce complementary literals (so that inferences become possible).
- Next idea: General resolution through lazy instantiation
  - Do not instantiate more than necessary to get complementary lits.

Refined derivation:

$i.$	$\neg Q(a) \vee \neg P(a)$		3.	$P(f(y))$	$ii\{x/f(y)\}$
$ii.$	$P(x)$		4.	$Q(y)$	$(iii, 3)$
$iii.$	$\neg P(f(y)) \vee Q(y)$		5.	$Q(a)$	$4\{y/a\}$
1.	$P(a)$	$ii\{x/a\}$	6.	$\bot$	$(2, 5)$
2.	$\neg Q(a)$	$(i, 1)$			

How to lift ground saturation to general clauses

- How can the idea of general resolution through lazy instantiation of complementary literals, be made effective and efficient?
- Challenge: Make saturation of infinite sets of instantiations of finitely many general clauses with variables effective and efficient.

- Solution due to Robinson (1965):
  - Use unification to find complementary literals (rather than syntactic identity).
  - Use only minimal unifiers (most general unifiers).
  - Unifiers do not need to be ground unifiers.

Advantage of using unification

- The advantage of the method in Robinson (1965) compared ground resolution exploiting Herbrand's theorem is that unification enumerates only those instances of clauses that participate in inferences.
- Moreover, clauses are not right away instantiated into ground clauses. Rather they are instantiated only as far as required for an inference.
- Inferences with non-ground clauses in general represent infinite sets of ground inferences which are computed simultaneously in a single step.

Unifiers

- Let
$$E = \{s_1 \doteq t_1, \ldots, s_n \doteq t_n\}$$
be a multi-set of equality problems, where s_i and t_i denote terms or atoms.
- A substitution $\boldsymbol{\sigma}$ is called a unifier of E, if $s_i\boldsymbol{\sigma} = t_i\boldsymbol{\sigma}$ for each $1 \leq i \leq n$.
- If a unifier of E exists, then E is said to be unifiable.
- Two expressions s and t are unifiable if $\{s \doteq t\}$ is unifiable, i.e. there is a substitution $\boldsymbol{\sigma}$ s.t. $s\boldsymbol{\sigma} = t\boldsymbol{\sigma}$.
- If $n > 1$, $\boldsymbol{\sigma}$ is said to a simultaneous unifier of all the pairs $s_i \doteq t_i$ in E.

Most general unifiers

- If $\boldsymbol{\sigma}$ unifies E then $\boldsymbol{\sigma\rho}$ also unifies E, for *any* substitution $\boldsymbol{\rho}$.
- $\boldsymbol{\sigma\rho}$ denotes the composition of $\boldsymbol{\sigma}$ and $\boldsymbol{\rho}$ as mappings, i.e.

$$
\begin{array}{lrll}
 & (\boldsymbol{\sigma\rho})(x) & =^{\text{def}} (x\boldsymbol{\sigma})\boldsymbol{\rho} & \text{for any } x \in \mathcal{X} \\
\text{In general:} & s(\boldsymbol{\sigma\rho}) & = (s\boldsymbol{\sigma})\boldsymbol{\rho} & \text{for any term } s \\
 & F(\boldsymbol{\sigma\rho}) & = (F\boldsymbol{\sigma})\boldsymbol{\rho} & \text{for any formula } F
\end{array}
$$

- If $\boldsymbol{\sigma}$ is a unifier of E and for any other unifier $\boldsymbol{\theta}$ of E, there is a substitution $\boldsymbol{\rho}$ such that $\boldsymbol{\sigma\rho} = \boldsymbol{\theta}$ then $\boldsymbol{\sigma}$ is a most general unifier of E. $\boldsymbol{\sigma}$ is then denoted by $\text{mgu}(E)$.
- Notation: $\text{mgu}(A, B)$ for $\text{mgu}(\{A \doteq B\})$
$\text{mgu}(A_1, \ldots, A_n)$ for $\text{mgu}(\{A_1 \doteq A_2, \ldots, A_1 \doteq A_n\})$

Unification theorem

Property 20 (Robinson)
Every unifiable system E has a most general unifier.

Proof: The unification algorithm (defined on the next slide) provides a method that, applied to input E, always terminates either with a solved form for E from which the mgu can be immediately read off, or with $\perp$ to indicate that E is not unifiable. This is shown in Lemma 21 below.

- A system E is in solved form, if $E = \{x_1 \doteq s_1, \ldots, x_k \doteq s_k\}$ with the x_i pairwise distinct, and $x_i \notin var(s_j)$.
- In this case E represents a unique (idempotent) substitution $\sigma_E = \{x_1/s_1, \ldots, x_k/s_k\}$.

A basic unification algorithm, based on rules

- Input: Set E of equational problems
- Goal: Determine if E is unifiable, and if it is, to read off mgu
- Output: Set E of equational problems in solved form or
  return $\perp$ (for *not unifiable*)
- We formalise the unification algorithm by an inference system based on $\Rightarrow_U$-unification rules
- Notation: $s \doteq t, E$ for $\{s \doteq t\} \cup E$
- Idea of each rule application:
  Pick an $s \doteq t$ in E, and try to unify s and t, and bring the entire set into solved form.

$\Rightarrow_U$-unification rules

Orientation: $t \doteq x,\ E \ \Rightarrow_U\ x \doteq t,\ E$

if $t \not\in \mathcal{X}$

Trivial: $t \doteq t,\ E \ \Rightarrow_U\ E$

Disagreement/Clash:

$$f(\ldots) \doteq g(\ldots),\ E \ \Rightarrow_U\ \perp$$

Decomposition:

$$f(s_1, \ldots, s_n) \doteq f(t_1, \ldots, t_n),\ E \ \Rightarrow_U\ s_1 \doteq t_1,\ \ldots,\ s_n \doteq t_n,\ E$$

Occur-check: $x \doteq t,\ E \ \Rightarrow_U\ \perp$

if $x \in var(t),\ x \neq t$

Substitution: $x \doteq t,\ E \ \Rightarrow_U\ x \doteq t,\ E\{x/t\}$

if $x \in var(E),\ x \not\in var(t)$

Examples

- Consider:

$$f(x, g(y)) \doteq f(g(w), g(z))$$
$$\Rightarrow_U \quad x \doteq g(w),\ g(y) \doteq g(z) \qquad \text{by Decomp.}$$
$$\Rightarrow_U \quad x \doteq g(w),\ y \doteq z \qquad \text{by Decomp.}$$

The most general unifier of $f(x, g(y))$ and $f(g(w), g(z))$ is

$$\sigma = \{x/g(w),\ y/z\}$$

- However $f(x, g(y))$ and $f(g(w), h(z))$ are not unifiable:

$$f(x, g(y)) \doteq f(g(w), h(z))$$
$$\Rightarrow_U \quad x \doteq g(w),\ g(y) \doteq h(z) \qquad \text{by Decomp.}$$
$$\Rightarrow_U \quad \perp \qquad \text{by Disagreem.}$$

Exercise

Compute the most general unifier of $P(f(z, g(a, y)), h(z))$ and $P(f(f(u, v), w), h(f(a, b)))$. Here u, v, w denote variables.

Correctness of $\Rightarrow_U$-unification

Lemma 21

(i) $\Rightarrow_U$ is well-founded, i.e. there is no infinite derivation
$E_0 \Rightarrow_U E_1 \Rightarrow_U E_2 \Rightarrow_U \ldots$

(ii) Let $E_0 \Rightarrow_U E_1 \Rightarrow_U \ldots \Rightarrow_U E_n$ be such that E_n is irreducible wrt. $\Rightarrow_U$.

a. If E_0 is unifiable then E_n is in solved form and $\boldsymbol{\sigma}_{E_n}$ is an mgu of E_0.

b. If E_0 is not unifiable then $E_n = \perp$.

Previously ...

- ground resolution for FO clauses based on ground instantiation and Herbrand's theorem
- refined idea: use unification to get complementary literals
- unification algorithm for computing most general unifiers

Basic resolution calculus for general clauses

- **General binary resolution calculus *Res*:**

$$\frac{C \vee A \qquad \neg B \vee D}{(C \vee D)\boldsymbol{\sigma}} \quad \text{if } \boldsymbol{\sigma} = \text{mgu}(A, B) \quad (\text{resolution})$$

$$\frac{C \vee A \vee B}{(C \vee A)\boldsymbol{\sigma}} \quad \text{if } \boldsymbol{\sigma} = \text{mgu}(A, B) \quad (\text{positive factoring})$$

- Important assumption for resolution rule:
  - Apply the resolution rule only to variable-disjoint clauses. If the premises share variables then first (bijectively) rename the variables such that the premises become variable-disjoint.
  - We do not formalise this. Which names one uses for variables is not relevant.

Sample derivation using general basic resolution *Res*

1.	$\neg Q(a) \vee \neg P(a)$	given
2.	$P(x)$	given
3.	$\neg P(f(y)) \vee Q(y)$	given
4.	$\neg Q(a)$	$(Res, 1, 2), \sigma = \{x/a\}$
5.	$Q(y)$	$(Res, 2, 3), \sigma = \{x/f(y)\}$
6.	$\perp$	$(Res, 4, 5), \sigma = \{y/a\}$

Lifting lemma

Lemma 22

Let C and D be variable-disjoint clauses. Let $C\sigma$ and $D\rho$ be any ground instances of C and D. If R is the resolvent of $C\sigma$ and $D\rho$, then there is a resolvent R' of C and D such that R is a ground instance of R'.

general resolution

$$\dfrac{\begin{array}{c} C \\ \sigma\,\vdots \\ C\sigma\end{array}\qquad\qquad \begin{array}{c} D \\ \vdots\,\rho \\ D\rho\end{array}}{R} \qquad \text{implies} \qquad \begin{array}{c}\dfrac{C \qquad\qquad D}{\exists R'} \\ \vdots\,\exists\theta \\ R = R'\theta\end{array}$$

propositional resolution

Lifting lemma (cont'd) & lifting saturation

- An analogous lifting lemma holds for factoring.
- The lifting lemmas imply that every inference step at ground level on instances of general clauses is an instance of an inference step at general level on these general clauses.

Corollary 23
Let N be a set of general clauses saturated under Res, i.e. $Res(N) \subseteq N$. Then also $G_\Sigma(N)$ is saturated, that is,

$$Res(G_\Sigma(N)) \subseteq G_\Sigma(N).$$

Proof: Consequence of the lifting lemmas.

Soundness and ref. completeness of general resolution

Property 24
Let N be a set of general clauses where $Res(N) \subseteq N$. Then

$$N \models \bot \;\text{ iff }\; \bot \in N.$$

Proof:
Let $Res(N) \subseteq N$. By Corollary 23: $Res(G_\Sigma(N)) \subseteq G_\Sigma(N)$.

$$\begin{aligned} N \models \bot \;\text{ iff }\; & G_\Sigma(N) \models \bot && \text{(Herbrand's theorem)} \\ \text{iff }\; & \bot \in G_\Sigma(N) && \text{(prop. resol. is refutat. complete)} \\ \text{iff }\; & \bot \in N \end{aligned}$$

Redundancy

- The ordering $\succ$ and the selection function S limit inferences to certain literals in clauses. They provide local restrictions of the rules in the resolution calculus.
- What about not performing inferences with clauses altogether? Is it possible to just delete clauses?
- Under which circumstances are clauses unnecessary?
- Goal: To introduce a general notion of redundancy that gives justification to tautology deletion, subsumption deletion and other techniques to eliminate redundant clauses.
- Intuitively a redundant clause is clause not needed for inference.

A formal notion of redundancy

- Let N be a set of ground clauses and C a ground clause (not necessarily in N). C is called redundant wrt. N, if there exist $C_1, \ldots, C_n \in N$, $n \geq 0$, such that
  (i) all $C_i \prec C$, and
  (ii) $C_1, \ldots, C_n \models C$.
- A general clause is redundant wrt. N if each ground instance $C\sigma$ of C either belongs to $G_\Sigma(N)$ or is redundant wrt. $G_\Sigma(N)$.
- C is redundant in N iff C is redundant wrt. $N \setminus \{C\}$.
- Idea: Redundant clauses are neither minimal exceptions nor productive. Note, the converse is not always true.
- Note: The same ordering $\succ$ is used for ordering restrictions and for redundancy (and for the completeness proof).

Examples of redundancy

Property 26

(i) If C tautology (i.e. $\models C$) then C is redundant wrt. any set N

(ii) If $C\boldsymbol{\sigma} \subset D$ then D is redundant wrt. $N \cup \{C\}$

(iii) If $C\boldsymbol{\sigma} \subseteq D$ then $D \vee \overline{L}\boldsymbol{\sigma}$ is redundant wrt. $N \cup \{C \vee L,\ D\}$, where $\overline{L}$ denotes the complement of L

- When $C\boldsymbol{\sigma} \subset D$ for some $\boldsymbol{\sigma}$ we say that D is strictly subsumed by C. (Under certain conditions one may also use non-strict subsumption, but this requires a slightly more complicated definition of redundancy.)
- $D \vee \overline{L}\boldsymbol{\sigma}$ is redundant wrt. any set containing D.

  If $C\boldsymbol{\sigma} \subseteq D$ then D is a 'repeated factor' of the resolvent of
  $$C \vee L \text{ and } D \vee \overline{L}\boldsymbol{\sigma}$$

Saturation up to redundancy

- Let $Red(N)$ denote the set of clauses redundant wrt. N.
- Recall, N is saturated wrt. $Res^{\succ}_{S}$ iff $Res^{\succ}_{S}(N) \subseteq N$.
- N is called saturated up to redundancy wrt. $Res^{\succ}_{S}$ iff

$$Res^{\succ}_{S}(N \setminus Red(N)) \subseteq N \cup Red(N)$$

  In words: every conclusion of an $Res^{\succ}_{S}$-inference with non-redundant clauses in N is in N or is redundant.

$Res_S^{\succ}$ up to redundancy is sound & refutat. complete

Property 27

Let N be saturated up to redundancy wrt. $Res_S^{\succ}$. Then

$$N \models \bot \ \text{ iff } \ \bot \in N.$$

Proof (Sketch):

Ground case:

- consider construction of candidate model $I_N^{\succ}$ for $Res_S^{\succ}$
- redundant clauses in N are not minimal exceptions for $I_N^{\succ}$
- redundant clauses are not productive

The premises of "essential" inferences are either minimal exceptions or productive.

Preservation/monotonicity properties of redundancy

Property 28

(i) $Red(N) \subseteq Red(M)$, if $N \subseteq M$

(ii) $Red(N) \subseteq Red(N \setminus M)$, if $M \subseteq Red(N)$

Proof: Exercise.

- This says that redundancy is preserved when, during a theorem proving process,
  (i) one adds (derives) new clauses or
  (ii) one deletes redundant clauses.

Rules for simplifications and deletion

- Some examples of standard simplification and deletion rules in provers:
  - Deletion of tautologies

$$N \cup \{C \vee A \vee \neg A\} \;\Rightarrow\; N$$

  - Deletion of subsumed clauses

$$N \cup \{C, D\} \;\Rightarrow\; N \cup \{C\}$$

    if $C\sigma \subseteq D$ (C subsumes D).

  - Reduction (also called subsumption resolution)

$$N \cup \{D \vee L,\; C \vee D\sigma \vee \overline{L}\sigma\} \;\Rightarrow\; N \cup \{D \vee L,\; C \vee D\sigma\}$$

$$N \cup \{C \vee L,\; D \vee C\sigma \vee \overline{L}\sigma\} \;\Rightarrow\; N \cup \{C \vee L,\; D \vee C\sigma\}$$

Summary

- general notion of redundancy
  - justifies deletion of clauses
  - standard instances:
    tautology deletion, strict subsumption deletion, subsumption resolution
- saturation up to redundancy
- soundness & completeness of $Res_S^{\succ}$ modulo redundancy

Summary: Resolution

- Resolution is a machine calculus
- Nevertheless it is the most powerful deduction calculus available
- All important principle:

  Avoid unnecessary inferences whenever possible

- Local restrictions of inferences via ground atom ordering $\succ$ and selection function S

  $\Rightarrow$ fewer inferences, fewer proof variants

  $\Rightarrow$ justification for numerous standard refinements

  $\Rightarrow$ termination on many decidable fragments

  $\Rightarrow$ simulation of certain tableau-based deduction methods and other deduction methods; synthesis of these

Summary: Resolution (cont'd)

- Global restrictions of search space via redundancy elimination

  $\Rightarrow$ delete clause, limit inferences to non-redundant clauses

  $\Rightarrow$ computing with "smaller" clause set, improves efficiency

- Further specialisation of inference systems required for reasoning with equality, specific algebraic theories (lattices, abelian groups, rings, fields), integers

Basic Notions

Let $\star$ be an operator defined over (elements of) a set X.

- $\star$ is commutative iff
  for any $x, y \in X$, $x \star y = y \star x$.
- $\star$ is associative iff
  for any $x, y, z \in X$, $((x \star y) \star z) = (x \star (y \star z))$.

Notation

- p, q, P, Q predicate symbols
- x, y, z variables
- a, b, c constants
- f, g function symbols
- s, t terms
- A, B atoms
- L literal
- $\overline{L}$ complement of literal L
- C, D clauses
- N set of clauses
- F, G formulae
- σ, θ, ρ substitutions
- x/s substitution of s for x
- Σ given signature
- $\mathcal{X}$ given set of variables
- T_Σ, $T_\Sigma(\mathcal{X})$ terms over sign.
- $\mathcal{I}$ f.o. interpretation
- β variable assignment
- $\mathcal{I}_\beta$ assignment to terms / formulae
- $\beta[x \mapsto u]$, $\mathcal{I}_\beta[x \mapsto u]$ modification of assignment
- I Herbrand interpretation
- S selection function
- $\succ$ ordering, $>$ precedence