Today's Lecture

- Normal form transformation
- Tableau calculus for first-order logic
- Tableau calculus for description logics

Definitory Normal Form Transformation

Let F be a formula.

1. Introduce a name for every sub-formula of F.
2. Write down the equivalence of name and sub-formula.
3. Replace outermost sub-formulas by their names.
4. Let D be the conjunction of these equivalences and P the name for F.
5. Transform $D \rightarrow P$ into normal form as usual.

Example (I)

1. $F = \neg(X \vee Y) \wedge Z$

2. Definitions:

$$P \leftrightarrow \neg(X \vee Y) \wedge Z$$

$$Q \leftrightarrow \neg(X \vee Y)$$

$$R \leftrightarrow (X \vee Y)$$

3. Replacements:

$$P \leftrightarrow Q \wedge Z$$

$$Q \leftrightarrow \neg R$$

$$R \leftrightarrow (X \vee Y)$$

Example (I)

4. Implication $D \rightarrow P$:

$$[P \leftrightarrow Q \wedge Z] \wedge$$

$$[Q \leftrightarrow \neg R] \wedge$$

$$[R \leftrightarrow (X \vee Y)] \rightarrow P$$

Example (I)

5. Normal Form:

$$[P \wedge \neg Q] \vee$$

$$[P \wedge \neg Z] \vee$$

$$[\neg P \wedge Q \wedge Z] \vee$$

$$[Q \wedge R] \vee$$

$$[\neg Q \wedge \neg R] \vee$$

$$[R \wedge \neg X \wedge \neg Y] \vee$$

$$[\neg R \wedge X] \vee$$

$$[\neg R \wedge Y] \vee$$

$$P$$

Properties of Definitional Normal Form

Theorem.

Let F' be the definitional DNF of formula F, then F valid iff F' valid.

The size of the formula increases by a constant factor only.

Example (II)

$(X_1 \vee Y_1) \wedge ... \wedge (X_n \vee Y_n)$

DNF without definitions:

$$[X_1 \wedge X_2 \wedge ... \wedge X_{n\text{-}1} \wedge X_n] \vee$$
$$[X_1 \wedge X_2 \wedge ... \wedge X_{n\text{-}1} \wedge Y_n] \vee$$
$$\vee ... \vee$$
$$[Y_1 \wedge Y_2 \wedge ... \wedge Y_{n\text{-}1} \wedge Y_n]$$

2^n clauses with n literals

Definitional DNF of $(X_1 \vee Y_1) \wedge \ldots \wedge (X_n \vee Y_n)$

Definitional DNF (prior to final step):

$$[P_1 \leftrightarrow Q_1 \wedge P_2] \wedge$$

$$[P_2 \leftrightarrow Q_2 \wedge P_3] \wedge$$

$$\wedge \ldots \wedge$$

$$[P_{n\text{-}1} \leftrightarrow Q_{n\text{-}1} \wedge Q_n] \wedge$$

$$[Q_1 \leftrightarrow X_1 \vee Y_1] \wedge$$

$$\wedge \ldots \wedge$$

$$[Q_n \leftrightarrow X_n \vee Y_n] \rightarrow$$

$$P_1$$

$6n - 2$ clauses with 2 or 3 literals

Skolem Normal Form

Formula F is in Skolem conjunctive normal form (CNF)

$:\Leftrightarrow$

F is of the form $(\forall \vec{x}_1)\, F_1 \wedge \ldots \wedge (\forall \vec{x}_n)\, F_n$ where

- F_i is a disjunction of literals
- $\vec{x}_i$ are the only variables in F_i

Skolemization

Proof by contradiction

$\models F$ iff $\neg F$ is inconsistent

Skolem's Theorem.

A formula $(\forall x_1, ..., x_n)(\exists y)\ F$ is inconsistent iff

$(\forall x_1, ..., x_n)\ F\{y / f(x_1, ..., x_n)\}$ is inconsistent,

where f does not occur in F

$f :\Leftrightarrow$ Skolem function

Tableau Calculus

Tableau :$\Leftrightarrow$ tree of sets of Skolem CNF formulas

1. $\{A \wedge B\} \cup \Phi$ can have child $\{A, B\} \cup \Phi$
2. $\{A \vee B\} \cup \Phi$ can have children $\{A\} \cup \Phi$ and $\{B\} \cup \Phi$
3. $\{(\forall x)\, A\} \cup \Phi$ can have child $\{A\{x/t\}, (\forall x)\, A\} \cup \Phi$ (t ground)

Tableau closed:$\Leftrightarrow$ $A, \neg A \in \Phi$ for every leaf Φ

(for some atom A)

Theorem.

Skolem CNF F inconsistent iff F root of some closed tableau

A Tableau Prover in Prolog (I)

```
prove((A,B), Phi, Lits, FreeV, VarLim) :- !,     % A and B
     prove(A, [B|Phi], Lits, FreeV, VarLim).

prove((A;B), Phi, Lits, FreeV, VarLim) :- !,     % A or B
     prove(A, Phi, Lits, FreeV, VarLim),
     prove(B, Phi, Lits, FreeV, VarLim).

prove(all(X,A), Phi, Lits, FreeV, VarLim) :- !,  % A(x)
     \+ length(FreeV, VarLim),
     copy_term((X,A,FreeV), (X1,A1,FreeV)),
     append(Phi, [all(X,A)], Phi1),
     prove(A1, Phi1, Lits, [X1|FreeV], VarLim).
```

A Tableau Prover in Prolog (II)

```
% unify(S,T) unifies S and T with occurs check

prove(L, _, [Lit|Lits], _, _) :-                    % leaf
    (L = -N; -L = N) -> ( unify(N, Lit) ;
                          prove(L, [], Lits, _, _) ).

prove(L, [F|Phi], Lits, FreeV, VarLim) :-
    prove(F, Phi, [L|Lits], FreeV, VarLim).

prove(F, VarLim) :- prove(F, [], [], [], VarLim).
```

Examples (I)

```
?- prove((all(X,p(X,a)), all(Y,-p(b,Y))), 1).

No

?- prove((all(X,p(X,a)), all(Y,-p(b,Y))), 2).

Yes


?- prove((all(X,p(X,f(X))), all(Y,-p(g(Y),Y))), 10).

No
```

Examples (II)

```
?- prove((all(X,(-p(X);q(X))),
         q(a), r(a),
         (all(Y,(-p(Y);-r(Y))))), 2).

No

?- prove((all(X,(-p(X);q(X))),
         p(a), r(a),
         (all(Y,(-q(Y);-r(Y))))), 2).

Yes
```

Description Logics: Basic Definitions

concept names :$\Leftrightarrow$ unary predicate symbols N_C

concept roles :$\Leftrightarrow$ binary predicate symbols N_R

$\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ interpretation :$\Leftrightarrow$

- $\Delta^{\mathcal{I}}$ non-empty set of individuals
- $P^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$, for every $P \in N_C$
- $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$, for every $r \in N_R$

Description Logic $\mathcal{ALC}$

Constructor	Syntax	Semantics
negation	$\neg C$	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
conjunction	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
disjunction	$C \sqcup D$	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$
existential restr.	$\exists r.\ C$	$\{x \in \Delta^{\mathcal{I}} \mid \exists y : (x, y) \in r^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$
value restr.	$\forall r.\ C$	$\{x \in \Delta^{\mathcal{I}} \mid \forall y : (x, y) \in r^{\mathcal{I}} \rightarrow y \in C^{\mathcal{I}}\}$

Example

Woman	≡ Human ⊓ Female
Man	≡ Human ⊓ ¬Female
Mother	≡ Woman ⊓ ∃hasChild. Human
Father	≡ Man ⊓ ∃hasChild. Human
Parent	≡ Mother ⊔ Father
GrandParent	≡ Parent ⊓ ∃hasChild. Parent
ProudParent	≡ Parent ⊓ ∀hasChild. Happy

Subsumption and Satisfiability

Concept description D subsumes concept description C

$:\Leftrightarrow$ $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$, for all interpretations $\mathcal{I}$

Concept description C and D equivalent

$:\Leftrightarrow$ C subsumes D and vice versa

Concept description C satisfiable

$:\Leftrightarrow$ $C^{\mathcal{I}} \neq \emptyset$, for some interpretation $\mathcal{I}$

ABox

Let N_I be a set of individual names.

ABox :$\Leftrightarrow$ finite set of assertions of the form

- $C(a)$ concept assertion
- $r(a, b)$ role assertion

where $a, b \in N_I$, C concept description, r role name.

Model for an ABox :$\Leftrightarrow$ interpretation $\mathcal{I}$ such that

- $\mathcal{I}$ additionally assigns elements $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$ to all $a \in N_I$
- $a^{\mathcal{I}} \in C^{\mathcal{I}}$, for every concept assertion $C(a)$
- $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in r^{\mathcal{I}}$, for every role assertion $r(a, b)$

Problem Reduction

1. D subsumes C iff $C \sqcap \neg D$ unsatisfiable

2. C satisfiable iff ABox $\{C(a)\}$ has a model

A Tableau Algorithm for Description Logics

1. Start with ABox $\mathcal{A}_0 = \{C(x_0)\}$

2. Apply transformation rules to obtain a maximal sequence

$$\mathcal{A}_0 \mapsto \mathcal{A}_1 \mapsto \ldots \mapsto \mathcal{A}_n$$

3. If $\mathcal{A}_n$ contains no pair $P(x)$, $\neg P(x)$ then C is satisfiable.

Transformation Rules for $\mathcal{ALC}$

$\mathcal{A}$ contains $(C_1 \sqcap C_2)(x)$ but not both $C_1(x)$ and $C_2(x)$

$\mapsto \mathcal{A}' := \mathcal{A} \cup \{C_1(x), C_2(x)\}$

$\mathcal{A}$ contains $(C_1 \sqcup C_2)(x)$ but neither $C_1(x)$ nor $C_2(x)$

$\mapsto \mathcal{A}' := \mathcal{A} \cup \{C_1(x)\}$, $\mathcal{A}'' := \mathcal{A} \cup \{C_2(x)\}$

$\mathcal{A}$ contains $(\exists r.C)(x)$ but not both $r(x, z)$ and $C(z)$ for some z

$\mapsto \mathcal{A}' := \mathcal{A} \cup \{C(y), r(x, y)\}$ for some new individual y

$\mathcal{A}$ contains $(\forall r.C)(x)$ and $r(x, y)$ but not $C(y)$

$\mapsto \mathcal{A}' := \mathcal{A} \cup \{C(y)\}$

Description Logic Tableaux

ABox complete :⇔ no transformation rule applies

ABox closed :⇔ contains $P(x)$, $\neg P(x)$ (P concept name, x individual)

Correctness of Decision Procedure

Theorem.

1. There cannot be an infinite sequence of rule applications

$$\mathcal{A}_0 = \{C(x_0)\} \mapsto \mathcal{A}_1 \mapsto \mathcal{A}_2 \mapsto \ldots$$

2a. If $\mathcal{A} \mapsto \mathcal{A}'$ then $\mathcal{A}$ satisfiable iff $\mathcal{A}'$ satisfiable.

2b. If $\mathcal{A} \mapsto \mathcal{A}'; \mathcal{A}''$ then $\mathcal{A}$ satisfiable iff $\mathcal{A}'$ or $\mathcal{A}''$ satisfiable.

3. Any closed ABox is inconsistent.

4. Any complete and open ABox is consistent.

DL Reasoning by Constraint Solving

Constraint Handling Rules (CHRs):

simplification rule: $H \Leftrightarrow B$
propagation rule: $H \Longrightarrow B$

Head H and body B are constraints.

CHR program :$\Leftrightarrow$ finite set of CHRs

CHRs: Operational Semantics

$\langle G; C\rangle$ state $:\Leftrightarrow$ G goal (sequence of atoms), C constraint (store)

initial state $:\Leftrightarrow \langle G; \mathit{true}\rangle$

successful final state $:\Leftrightarrow \langle \square ; C\rangle$

failed final state $:\Leftrightarrow \langle G; \mathit{false}\rangle$

if F is a constraint, then $\langle F, G; C\rangle \mapsto \langle G; F \wedge C\rangle$

CHR Transition Rule: Simplify

Simplify

If $H \Leftrightarrow B$ is a fresh variant of a CHR with variables $\vec{x}$

And $\models (\forall)(C \rightarrow (\exists \vec{x})\ F \doteq H)$

Then $\langle G;\ \mathrm{F} \wedge\ C\rangle \mapsto \langle \mathrm{B},\ G;\ (F \doteq H) \wedge C\rangle$

CHR Transition Rule: Propagate

Propagate

If $H \Longrightarrow B$ is a fresh variant of a CHR with variables $\vec{x}$

And $\models (\forall)(C \rightarrow (\exists \vec{x})\ F \doteq H)$

Then $\langle G;\ \mathrm{F} \wedge C\rangle \mapsto \langle \mathrm{B},\ G;\ \mathrm{F} \wedge (F \doteq H) \wedge C\rangle$

Never apply the same rule twice to the same constraint.

CHRs for $\mathcal{ALC}$ Reasoning

`I:s` `I` variable or individual, `s` concept description

`(I,J):r` `I,J` variables or individuals, `r` role name

```
I:S1 and S2      <=> I:S1, I:S2
I:S1 or S2       <=> I:S1 ; I:S2.
I:some R is S    <=> (I,J):R, J:S.
I:every R is S,
(I,J):R          ==> J:S.
```

Handling Negation, Clash, and Definitions

```
I:not not S           <=> I:S.
I:not (S1 and S2)     <=> I:not S1 or not S2.
I:not (S1 or S2)      <=> I:not S1 and not S2.
I:not (some R is S)   <=> I:every R is not S.
I:not (every R is S)  <=> I:some R is not S.

I:not S, I:S <=> false.
```

For every concept definition $C \equiv S$,

```
I:C       <=> I:S.
I: not C  <=> I:not S.
```

Example

```
I:woman          <=> I:human and female.
I:man            <=> I:human and not female.
I:mother         <=> I:woman and some hasChild is human.
I:father         <=> I:man and some hasChild is human.
I:parent         <=> I:mother or father.
I:grandParent    <=> I:parent and some hasChild is parent.
I:proudParent    <=> I:parent and every hasChild is happy.

I:not woman      <=> I:not (human and female).
...
```

Example Query (I)

```
?- sue:proudParent, (sue,joe):hasChild, joe:not happy.
```

Unfold:

```
sue:parent and every hasChild is happy, ...
```

Unfold and split ⊓:

```
..., sue:some hasChild is human, sue:every hasChild is happy, ...
```

Simplify existence restriction:

```
..., (sue,X):hasChild, X:human, sue:every hasChild is happy, ...
```

Propagate value restriction:

```
..., X:happy, joe:happy, (sue,X):hasChild, X:human,
sue:every hasChild is happy, (sue,joe):hasChild, joe:not happy
```

Clash:

```
false
```

Example Query (II)

```
?- a:grandParent and not parent

No.


?- a:grandParent and not mother.

a:human, a:not female, (a,X):hasChild, X:human,
X:female, (X,Y):hasChild, Y:human.
```

Objectives

- Normal form transformation
- Tableau calculus for first-order logic
- Tableau calculus for description logics