Non-classical logics

Lecture 18: Modal logics (Part 5)

Viorica Sofronie-Stokkermans

sofronie@uni-koblenz.de

Until now

- History and Motivation
- Propositional modal logic

    Syntax

    Inference systems and proofs

    Semantics (models, validity, satisfiability)

    Entailment (local/global); Deduction theorem

    Correspondence theory; First-order definability

    Proof Systems

        Inference system
        Tableau calculi
        Resolution

    Decidability

Decidability of modal logics

Decidability of modal logics

- **Direct approach:** Prove finite model property

  If a formula F is satisfiable then it has a model with at least $f(\textit{size}(F))$ elements, where f is a concrete function.

  Generate all models with $1, 2, 3, \ldots, f(\textit{size}(F))$ elements.

Decidability of modal logics

- **Direct approach:** Prove finite model property

  If a formula F is satisfiable then it has a model with at least $f(size(F))$ elements, where f is a concrete function.

  Generate all models with $1, 2, 3, \ldots, f(size(F))$ elements.

- **Alternative approaches:**
  - Show that terminating sound and complete tableau calculi exist
  - Show that ordered resolution ($+$ additional refinements) terminates on the type of first-order formulae which are generated starting from a modal formula.

Decidability

Idea:

We show that if a formula A has n subformulae, then
$\vdash_K A$ iff A is valid in all frames having at most 2^n elements.

or alternatively, that the following are equivalent:

(1) There exists a Kripke structure $\mathcal{K} = (S, R, I)$ and $s \in S$ such that $(\mathcal{K}, s) \models A$.

(2) There exists a Kripke structure $\mathcal{K}' = (S', R', I')$ and $s' \in S'$ such that:
- $(\mathcal{K}', s') \models A$
- S' consists of at most 2^n states.

Goal: Construct the finite Kripke structure $\mathcal{K}'$ starting from $\mathcal{K}$.

Decidability

Filtrations

Fix a model $\mathcal{K} = (S, R, I)$ and a set $\Gamma \subseteq Fma_\Sigma$ that is closed under subformulae, i.e. $B \in \Gamma$ implies Subformulae(B) $\subseteq \Gamma$.

For each $s \in S$, define $\Gamma_s = \{B \in \Gamma \mid (\mathcal{K}, s) \models B\}$

and put $s \sim_\Gamma t$ iff $\Gamma_s = \Gamma_t$,

Then $s \sim_\Gamma t$ iff for all $B \in \Gamma$, $(\mathcal{K}, s) \models B$ iff $(\mathcal{K}, t) \models B$.

Fact: $\sim_\Gamma$ is an equivalence relation on S.

Let $[s] = \{t \mid s \sim_\Gamma t\}$ be the $\sim_\Gamma$-equivalence class of s.

Let $S_\Gamma := \{[s] \mid s \in S\}$ be the set of all such equivalence classes.

Lemma. If Γ is finite, then S_Γ is finite and has at most 2^n elements, where n is the number of elements of Γ.

Decidability

Goal: $(\mathcal{K}, s) \models A \quad \mapsto \quad (\mathcal{K}', s') \models A, \mathcal{K}' = (S', R', I'), \quad |S'| \leq 2^n.$

Step 1: $S' := S_\Gamma$, where $\Gamma = \text{Subformulae}(S)$

Step 2: $I' : (\Pi \cap \Gamma) \times S' \to \{0, 1\}$ def. by $I'(P, [s]) = I(P, s)$

Step 3: R' def. e.g. by: $([s], [t]) \in R'$ iff $\exists s' \in [s], \exists t' \in [t]$: $(s', t') \in R$

Remark: R' has the following properties:

(F1) if $(s, t) \in R$ then $([s], [t]) \in R'$

(F2) if $([s], [t]) \in R'$ then for all B, if $\Box B \in \Gamma$ and $(\mathcal{K}, s) \models \Box B$, then $(\mathcal{K}, t) \models B$.

Any Kripke structure $\mathcal{K}' = (S_\Gamma, R', I_\Gamma)$ in which R' satisfies (Fl) and (F2) is called a Γ-filtration of $\mathcal{K}$.

Decidability

Examples of filtrations

- The smallest filtration.
  $([s],[t]) \in R'$ iff $\exists s' \sim_\Gamma s, \exists t' \sim_\Gamma t (s', t') \in R$.
- The largest filtration.
  $([s],[t]) \in R$ iff for all $B, \Box B \in \Gamma$, $(\mathcal{K}, s) \models \Box B$ implies $(\mathcal{K}, t) \models B$.
- The transitive filtration.
  $([s],[t]) \in R'$ iff for all $B, \Box B \in \Gamma$, $(\mathcal{K}, s) \models \Box B$ implies $(\mathcal{K}, t) \models \Box B \wedge B$.

When defining $\mathcal{K}'$ we can choose also the second or third definition of R'.

Decidability

Filtration Lemma.

Let Γ be a set of modal formulae closed under subformulae.
Let $\mathcal{K} = (S, R, I)$ be a Kripke structure and let $\mathcal{K}' = (S_\Gamma, R', I_\Gamma)$ be a Γ-filtration of $\mathcal{K}$.

If $B \in \Gamma$, then for any $s \in S$, $\quad (\mathcal{K}, s) \models B \quad$ iff $\quad (\mathcal{K}', [s]) \models B$

Proof. The case $B = P \in \Pi \cap \Gamma$ is given by the definition of I'

The inductive case for the connectives $\{\wedge, \vee, \neg\}$ is straightforward.

The inductive case for $\Box$ uses (FI) and (F2).

Note that the closure of Γ under subformulae is needed in order to be able to apply the induction hypothesis.

Decidability

Theorem. Let A be a formula with n subformulae.
Then $\vdash_K A$ iff A is valid in all frames having at most 2^n elements.

Proof. Suppose $\not\vdash_K A$. Then there is a model $\mathcal{K} = (S, R, I)$ and a state $s \in S$ at which A is false. Let $\Gamma = \text{Subformulae}(A)$.

Then Γ is closed under subformulae, so we can construct Γ-filtrations $\mathcal{K}' = (S_\Gamma, R', I_\Gamma)$ as above. By the Filtration Lemma, A is false at $[s]$ in any such model, and hence not valid in the frame (S_Γ, R').

We previously showed that the desired bound on the size of S_Γ is 2^n.

Decidability

A logic $\mathcal{L}$ characterized by a set $\mathcal{F}$ of frames* has the finite frame property if it is determined by its finite frames, i.e.,

if $\nvdash_{\mathcal{L}} A$, then there is a finite frame $F \in \mathcal{F}$ s.t. $\mathcal{F} \not\models A$

We showed that the smallest normal logic K has the finite frame property, and a computable bound was given on the size of the invalidating frame for a given non-theorem.

* We can choose $\mathcal{F}$ to be the class of all frames in which all theorems of $\mathcal{L}$ are valid.

Decidability

This implies that the property of K-theoremhood is decidable, i.e. that there is an algorithm for determining, for each formula A, whether or not $\vdash_K A$:

If A has n subformulae, we simply check to see whether or not A is valid in all frames of size at most 2^n.

- Since a finite set has finitely many binary relations (2^{m^2} relations on an m-element set), there are only finitely many frames of size at most 2^n.

- Moreover, to determine whether A is valid on a finite frame F, we need only look at models $I : \Pi \cap \mathsf{Subformulae}(A) \to \{0, 1\}$ on F.

But there are only finitely many such models on F. Thus the whole checking procedure for validity of A in frames of size at most 2^n can be completed in a finite amount of time.

Other modal systems

System	*Description*
T	$K + \Box A \to A$
D	$K + \Box A \to \Diamond A$
B	$T + \neg A \to \Box\neg\Box A$
$S4$	$T + \Box A \to \Box\Box A$
$S5$	$T + \neg\Box A \to \Box\neg\Box A$
$S4.2$	$S4 + \diamond\Box A \to \Box\Diamond A$
$S4.3$	$S4 + \Box(\Box(A \to B)) \vee \Box(\Box(B \to A))$
C	$K + \frac{A \to B}{\Box(A \to B)}$ instead of (G).

Other modal systems

We say that $\mathcal{L}$ (with characterizing class of frames $\mathcal{F}$ has the strong finite frame property if there is a computable function g such that

if $\nvdash_{\mathcal{L}} A$, then there is a finite frame $F \in \mathcal{F}$ that

- invalidates A and
- has at most $g(n)$ elements, where n is the number of subformulae of A.

In adapting the above decidability argument to $\mathcal{L}$, in addition to deciding whether or not a given finite frame F validates A, we also have to decide whether or not $F \in \mathcal{F}$.

If $\mathcal{L}$ is finitely axiomatisable, meaning that $\mathcal{L} = KS_l...S_n$ for some finite number of schemata, then $\mathcal{F}$ is the class of all frames in which the axioms schemata $S_1, \ldots, S_n$ hold.

Then the property "$F \in \mathcal{F}$" is decidable: it suffices to determine whether each S_j is valid in F.

Other modal systems

We say that $\mathcal{L}$ (with characterizing class of frames $\mathcal{F}$ has the strong finite frame property if there is a computable function g such that

if $\not\vdash_{\mathcal{L}} A$, then there is a finite frame $F \in \mathcal{F}$ that

- invalidates A and

- has at most $g(n)$ elements, where n is the number of subformulae of A.

Theorem. Every finitely axiomatisable propositional modal logic with the strong finite frame property is decidable.

Other modal systems

We say that $\mathcal{L}$ (with characterizing class of frames $\mathcal{F}$ has the strong finite frame property if there is a computable function g such that

if $\nvdash_{\mathcal{L}} A$, then there is a finite frame $F \in \mathcal{F}$ that

- invalidates A and

- has at most $g(n)$ elements, where n is the number of subformulae of A.

Theorem. Every finitely axiomatisable logic with the strong finite frame property is decidable.

In fact it can be shown that any finitely axiomatisable logic with the finite frame property is decidable.

Other modal systems

Remark: For many of the logics we have considered thus far, validity of S_j is equivalent to some first-order property of R, which can be algorithmically decided for finite F.

Examples

Axiom	Property of R
$\Box A \to A$	reflexive
$A \to \Box\Diamond A$	symmetric
$\Box A \to \Box\Box A$	transitive

Consequence: The extension of K with each of the axioms above is decidable.

Proof It is sufficient to show that if Γ-filtrations are as defined in this lecture:
- for any reflexive frame its Γ-filtration is again reflexive
- for any symmetric frame its Γ-filtration is again symmetric

Transitivity is not always preserved by the minimal Γ-filtration of R (which was the one we used when defining the finite model $\mathcal{K}'$); instead we can use the transitive filtration.

Decidability of modal logics

- **Direct approach:** Prove finite model property

  If a formula F is satisfiable then it has a model with at least $f(\mathit{size}(F))$ elements, where f is a concrete function.

  Generate all models with $1, 2, 3, \dots, f(\mathit{size}(F))$ elements.

- **Alternative approaches:**
  - Show that terminating sound and complete tableau calculi exist
  - Show that ordered resolution (+ additional refinements) terminates on the type of first-order formulae which are generated starting from a modal formula.

Decidability of modal logics

- **Direct approach:** Prove finite model property

  If a formula F is satisfiable then it has a model with at least $f(\mathit{size}(F))$ elements, where f is a concrete function.

  Generate all models with $1, 2, 3, \ldots, f(\mathit{size}(F))$ elements.

- **Alternative approaches:**
  - Show that terminating sound and complete tableau calculi exist
  - Show that ordered resolution (+ additional refinements) terminates on the type of first-order formulae which are generated starting from a modal formula.

Translation for classical logic

$\mathcal{K} = (S, R, I)$ Kripke model

$$
\begin{array}{rcll}
\mathrm{val}_{\mathcal{K}}(\bot)(s) & = & 0 & \text{for all } s \\
\mathrm{val}_{\mathcal{K}}(\top)(s) & = & 1 & \text{for all } s \\
\mathrm{val}_{\mathcal{K}}(P)(s) = 1 & \leftrightarrow & I(P)(s) = 1 & \text{for all } s \\
\mathrm{val}_{\mathcal{K}}(\neg F)(s) = 1 & \leftrightarrow & \mathrm{val}_{\mathcal{K}}(F)(s) = 0 & \text{for all } s \\
\mathrm{val}_{\mathcal{K}}(F_1 \wedge F_2)(s) = 1 & \leftrightarrow & \mathrm{val}_{\mathcal{K}}(F_1)(s) \wedge \mathrm{val}_{\mathcal{K}}(F_1)(s) = 1 & \text{for all } s \\
\mathrm{val}_{\mathcal{K}}(F_1 \vee F_2)(s) = 1 & \leftrightarrow & \mathrm{val}_{\mathcal{K}}(F_1)(s) \vee \mathrm{val}_{\mathcal{K}}(F_1)(s) = 1 & \text{for all } s \\
\mathrm{val}_{\mathcal{K}}(\Box F)(s) = 1 & \leftrightarrow & \forall s'(R(s, s') \rightarrow \mathrm{val}_{\mathcal{K}}(F)(s') = 1 & \text{for all } s \\
\mathrm{val}_{\mathcal{K}}(\Diamond F)(s) = 1 & \leftrightarrow & \exists s'(R(s, s') \text{ and } \mathrm{val}_{\mathcal{K}}(F)(s') = 1 & \text{for all } s
\end{array}
$$

Translation:

$$
\begin{array}{lcl}
P \in \Pi & \mapsto & P/1 \text{ unary predicate} \\
F \text{ formula} & \mapsto & P_F/1 \text{ unary predicate} \\
R \text{ acc.rel} & \mapsto & R/2 \text{ binary predicate} \\
\mathrm{val}_{\mathcal{K}}(P)(s) = 1 & \mapsto & P(s) \\
\mathrm{val}_{\mathcal{K}}(P)(s) = 0 & \mapsto & \neg P(s)
\end{array}
$$

$$
\begin{array}{rcl}
\forall s(P_{\neg F}(s) & \leftrightarrow & \neg P_F(s)) \\
\forall s(P_{F_1 \wedge F_2}(s) & \leftrightarrow & P_{F_1}(s) \wedge P_{F_2}(s)) \\
\forall s(P_{F_1 \vee F_2}(s) & \leftrightarrow & P_{F_1}(s) \vee P_{F_2}(s)) \\
\forall s(P_{\Box F}(s) & \leftrightarrow & \forall s'(R(s, s') \rightarrow P_F(s'))) \\
\forall s(P_{\Diamond F}(s) & \leftrightarrow & \exists s'(R(s, s') \wedge P_F(s')))
\end{array}
$$

Translation for classical logic

$\mathcal{K} = (S, R, I)$ Kripke model

$$
\begin{array}{rcll}
\mathrm{val}_{\mathcal{K}}(\bot)(s) & = & 0 & \text{for all } s \\
\mathrm{val}_{\mathcal{K}}(\top)(s) & = & 1 & \text{for all } s \\
\mathrm{val}_{\mathcal{K}}(P)(s) = 1 & \leftrightarrow & I(P)(s) = 1 & \text{for all } s \\
\mathrm{val}_{\mathcal{K}}(\neg F)(s) = 1 & \leftrightarrow & \mathrm{val}_{\mathcal{K}}(F)(s) = 0 & \text{for all } s \\
\mathrm{val}_{\mathcal{K}}(F_1 \wedge F_2)(s) = 1 & \leftrightarrow & \mathrm{val}_{\mathcal{K}}(F_1)(s) \wedge \mathrm{val}_{\mathcal{K}}(F_1)(s) = 1 & \text{for all } s \\
\mathrm{val}_{\mathcal{K}}(F_1 \vee F_2)(s) = 1 & \leftrightarrow & \mathrm{val}_{\mathcal{K}}(F_1)(s) \vee \mathrm{val}_{\mathcal{K}}(F_1)(s) = 1 & \text{for all } s \\
\mathrm{val}_{\mathcal{K}}(\Box F)(s) = 1 & \leftrightarrow & \forall s'(R(s, s') \rightarrow \mathrm{val}_{\mathcal{K}}(F)(s') = 1 & \text{for all } s \\
\mathrm{val}_{\mathcal{K}}(\Diamond F)(s) = 1 & \leftrightarrow & \exists s'(R(s, s') \text{ and } \mathrm{val}_{\mathcal{K}}(F)(s') = 1 & \text{for all } s
\end{array}
$$

Translation: Given F modal formula:

$$
\begin{array}{lcl}
P \in \Pi & \mapsto & P/1 \text{ unary predicate} \\
F' \text{ subformula of F} & \mapsto & P_F/1 \text{ unary predicate} \\
R \text{ acc.rel} & \mapsto & R/2 \text{ binary predicate} \\
\mathrm{val}_{\mathcal{K}}(P)(s) = 1 & \mapsto & P(s) \\
\mathrm{val}_{\mathcal{K}}(P)(s) = 0 & \mapsto & \neg P(s)
\end{array}
$$

$$
\begin{array}{rcl}
\forall s(P_{\neg F'}(s) & \leftrightarrow & \neg P_{F'}(s)) \\
\forall s(P_{F_1 \wedge F_2}(s) & \leftrightarrow & P_{F_1}(s) \wedge P_{F_2}(s)) \\
\forall s(P_{F_1 \vee F_2}(s) & \leftrightarrow & P_{F_1}(s) \vee P_{F_2}(s)) \\
\forall s(P_{\Box F'}(s) & \leftrightarrow & \forall s'(R(s, s') \rightarrow P_{F'}(s'))) \\
\forall s(P_{\Diamond F'}(s) & \leftrightarrow & \exists s'(R(s, s') \wedge P_{F'}(s')))
\end{array}
$$

where the index formulae range over all subfromulae of F.

Translation to classical logic

Translation: Given F modal formula:

$P \in \Pi$	$\mapsto$	$P/1$ unary predicate
F' subformula of F	$\mapsto$	$P_{F'}/1$ unary predicate
R acc.rel	$\mapsto$	$R/2$ binary predicate
$\mathrm{val}_{\mathcal{K}}(P)(s) = 1$	$\mapsto$	$P(s)$
$\mathrm{val}_{\mathcal{K}}(P)(s) = 0$	$\mapsto$	$\neg P(s)$

$$\underbrace{\begin{array}{rcl} \forall s(P_{\neg F'}(s) & \leftrightarrow & \neg P_{F'}(s)) \\ \forall s(P_{F_1 \wedge F_2}(s) & \leftrightarrow & P_{F_1}(s) \wedge P_{F_2}(s)) \\ \forall s(P_{F_1 \vee F_2}(s) & \leftrightarrow & P_{F_1}(s) \vee P_{F_2}(s)) \\ \forall s(P_{\Box F'}(s) & \leftrightarrow & \forall s'(R(s, s') \rightarrow P_{F'}(s'))) \\ \forall s(P_{\Diamond F'}(s) & \leftrightarrow & \exists s'(R(s, s') \wedge P_{F'}(s'))) \\ \multicolumn{3}{l}{\text{where the index formulae range over all subformulae of } F.} \end{array}}_{\mathrm{Rename}(F)}$$

Theorem.

F is K-satisfiable iff $\exists x P_F(x) \wedge \mathrm{Rename}(F)$ is satisfiable in first-order logic.

We now analyze the FO formula obtained

$$
\underbrace{
\begin{array}{ll}
\exists x & \neg P_F(x) \\[2ex]
\forall s & (P_{\neg F'}(s) \leftrightarrow \neg P_{F'}(s)) \\[2ex]
\forall s & (P_{F_1 \wedge F_2}(s) \leftrightarrow P_{F_1}(s) \wedge P_{F_2}(s)) \\[2ex]
\forall s & (P_{F_1 \vee F_2}(s) \leftrightarrow P_{F_1}(s) \vee P_{F_2}(s)) \\[2ex]
\forall s & (P_{\Box F'}(s) \leftrightarrow \forall s'(R(s,s') \rightarrow P_{F'}(s'))) \\[2ex]
\forall s & (P_{\Diamond F'}(s) \leftrightarrow \exists s'(R(s,s') \wedge P_{F'}(s'))) \\[2ex]
& \text{index formulae range over all subformulae of } F.
\end{array}
}_{\text{Rename}(F)}
$$

We now analyze the FO formula obtained

$$\exists x \qquad \neg P_F(x)$$

$$\underbrace{\begin{array}{lrcl}
\forall s & (P_{\neg F'}(s) & \leftarrow & \neg P_{F'}(s)) \\
\forall s & (P_{\neg F'}(s) & \rightarrow & \neg P_{F'}(s)) \\[1em]
\forall s & (P_{F_1 \wedge F_2}(s) & \leftarrow & P_{F_1}(s) \wedge P_{F_2}(s)) \\
\forall s & (P_{F_1 \wedge F_2}(s) & \rightarrow & P_{F_1}(s) \wedge P_{F_2}(s)) \\[1em]
\forall s & (P_{F_1 \vee F_2}(s) & \leftarrow & P_{F_1}(s) \vee P_{F_2}(s)) \\
\forall s & (P_{F_1 \vee F_2}(s) & \rightarrow & P_{F_1}(s) \vee P_{F_2}(s)) \\[1em]
\forall s & (P_{\Box F'}(s) & \leftarrow & \forall s'(R(s,s') \rightarrow P_{F'}(s'))) \\
\forall s & (P_{\Box F'}(s) & \rightarrow & \forall s'(R(s,s') \rightarrow P_{F'}(s'))) \\[1em]
\forall s & (P_{\Diamond F'}(s) & \leftarrow & \exists s'(R(s,s') \wedge P_{F'}(s'))) \\
\forall s & (P_{\Diamond F'}(s) & \rightarrow & \exists s'(R(s,s') \wedge P_{F'}(s'))) \\[1em]
\multicolumn{4}{c}{\text{index formulae range over all subformulae of } F.}
\end{array}}_{\text{Rename}(F)}$$

We now analyze the FO formula obtained

$$
\underbrace{
\begin{array}{lrcl}
\exists x & \neg P_F(x) & & \\
 & & & \\
\forall s & (P_{\neg F'}(s) & \vee & P_{F'}(s)) \\
\forall s & (\neg P_{\neg F'}(s) & \vee & \neg P_{F'}(s)) \\
 & & & \\
\forall s & (P_{F_1 \wedge F_2}(s) & \vee & \neg(P_{F_1}(s) \wedge P_{F_2}(s))) \\
\forall s & (\neg P_{F_1 \wedge F_2}(s) & \vee & P_{F_1}(s) \wedge P_{F_2}(s)) \\
 & & & \\
\forall s & (P_{F_1 \vee F_2}(s) & \vee & \neg(P_{F_1}(s) \vee P_{F_2}(s))) \\
\forall s & (\neg P_{F_1 \vee F_2}(s) & \vee & P_{F_1}(s) \vee P_{F_2}(s)) \\
 & & & \\
\forall s & (P_{\Box F'}(s) & \vee & \neg(\forall s'(R(s,s') \rightarrow P_{F'}(s')))) \\
\forall s & (\neg P_{\Box F'}(s) & \vee & \forall s'(R(s,s') \rightarrow P_{F'}(s'))) \\
 & & & \\
\forall s & (P_{\Diamond F'}(s) & \vee & \neg(\exists s'(R(s,s') \wedge P_{F'}(s')))) \\
\forall s & (\neg P_{\Diamond F'}(s) & \vee & \exists s'(R(s,s') \wedge P_{F'}(s'))) \\
 & \multicolumn{3}{l}{\text{index formulae range over all subformulae of } F.}
\end{array}
}_{\mathrm{Rename}(F)}
$$

We now analyze the FO formula obtained

$$
\underbrace{
\begin{array}{lrcl}
\exists x & \neg P_F(x) & & \\
 & & & \\
\forall s & (P_{\neg F'}(s) & \vee & P_{F'}(s)) \\
\forall s & (\neg P_{\neg F'}(s) & \vee & \neg P_{F'}(s)) \\
 & & & \\
\forall s & (P_{F_1 \wedge F_2}(s) & \vee & \neg(P_{F_1}(s) \wedge P_{F_2}(s))) \\
\forall s & (\neg P_{F_1 \wedge F_2}(s) & \vee & P_{F_1}(s) \wedge P_{F_2}(s)) \\
 & & & \\
\forall s & (P_{F_1 \vee F_2}(s) & \vee & \neg(P_{F_1}(s) \vee P_{F_2}(s))) \\
\forall s & (\neg P_{F_1 \vee F_2}(s) & \vee & P_{F_1}(s) \vee P_{F_2}(s)) \\
 & & & \\
\forall s & (P_{\Box F'}(s) & \vee & \exists s' \neg(R(s,s') \rightarrow P_{F'}(s'))) \\
\forall s & (\neg P_{\Box F'}(s) & \vee & \forall s'(R(s,s') \rightarrow P_{F'}(s'))) \\
 & & & \\
\forall s & (P_{\Diamond F'}(s) & \vee & \forall s' \neg(R(s,s') \wedge P_{F'}(s'))) \\
\forall s & (\neg P_{\Diamond F'}(s) & \vee & \exists s'(R(s,s') \wedge P_{F'}(s'))) \\
 & & & \\
\multicolumn{4}{c}{\text{index formulae range over all subformulae of } F.}
\end{array}
}_{\text{Rename}(F)}
$$

We now analyze the FO formula obtained

$$\underbrace{\begin{array}{llcl}
\exists x & \neg P_F(x) & & \\
 & & & \\
\forall s & (P_{\neg F'}(s) & \vee & P_{F'}(s)) \\
\forall s & (\neg P_{\neg F'}(s) & \vee & \neg P_{F'}(s)) \\
 & & & \\
\forall s & (P_{F_1 \wedge F_2}(s) & \vee & \neg(P_{F_1}(s) \wedge P_{F_2}(s))) \\
\forall s & (\neg P_{F_1 \wedge F_2}(s) & \vee & (P_{F_1}(s) \wedge P_{F_2}(s))) \\
 & & & \\
\forall s & (P_{F_1 \vee F_2}(s) & \vee & \neg(P_{F_1}(s) \vee P_{F_2}(s))) \\
\forall s & (\neg P_{F_1 \vee F_2}(s) & \vee & P_{F_1}(s) \vee P_{F_2}(s)) \\
 & & & \\
\forall s \exists s' & (P_{\Box F'}(s) & \vee & \neg(R(s,s') \rightarrow P_{F'}(s'))) \\
\forall s \forall s' & (\neg P_{\Box F'}(s) & \vee & (R(s,s') \rightarrow P_{F'}(s'))) \\
 & & & \\
\forall s \forall s' & (P_{\Diamond F'}(s) & \vee & \neg(R(s,s') \wedge P_{F'}(s'))) \\
\forall s \exists s' & (\neg P_{\Diamond F'}(s) & \vee & (R(s,s') \wedge P_{F'}(s'))) \\
\multicolumn{4}{c}{\text{index formulae range over all subformulae of } F.}
\end{array}}_{\mathrm{Rename}(F)}$$

Skolemization

$$\underbrace{\begin{array}{lrcl}
 & \neg P_F(c) & & \\[2em]
\forall s & (P_{\neg F'}(s) & \vee & P_{F'}(s)) \\
\forall s & (\neg P_{\neg F'}(s) & \vee & \neg P_{F'}(s)) \\[1em]
\forall s & (P_{F_1 \wedge F_2}(s) & \vee & \neg(P_{F_1}(s) \wedge P_{F_2}(s))) \\
\forall s & (\neg P_{F_1 \wedge F_2}(s) & \vee & (P_{F_1}(s) \wedge P_{F_2}(s))) \\[1em]
\forall s & (P_{F_1 \vee F_2}(s) & \vee & \neg(P_{F_1}(s) \vee P_{F_2}(s))) \\
\forall s & (\neg P_{F_1 \vee F_2}(s) & \vee & P_{F_1}(s) \vee P_{F_2}(s)) \\[1em]
\forall s & (P_{\Box F'}(s) & \vee & \neg(R(s, f_i(s)) \to P_{F'}(f_i(s)))) \\
\forall s \forall s' & (\neg P_{\Box F'}(s) & \vee & (R(s, s') \to P_{F'}(s'))) \\[1em]
\forall s \forall s' & (P_{\Diamond F'}(s) & \vee & \neg(R(s, s') \wedge P_{F'}(s'))) \\
\forall s & (\neg P_{\Diamond F'}(s) & \vee & (R(s, f_j(s)) \wedge P_{F'}(f_j(s)))) \\[1em]
\multicolumn{4}{c}{\text{index formulae range over all subformulae of } F.}
\end{array}}_{\text{Rename}(F)}$$

Translation to CNF

$$\underbrace{\begin{array}{lrcl}
 & \neg P_F(c) & & \\[1em]
\forall s & (P_{\neg F'}(s) & \vee & P_{F'}(s)) \\
\forall s & (\neg P_{\neg F'}(s) & \vee & \neg P_{F'}(s)) \\[0.5em]
\forall s & (P_{F_1 \wedge F_2}(s) & \vee & \neg P_{F_1}(s) \vee \neg P_{F_2}(s)) \\
\forall s & (\neg P_{F_1 \wedge F_2}(s) & \vee & (P_{F_1}(s) \wedge P_{F_2}(s))) \\[0.5em]
\forall s & (P_{F_1 \vee F_2}(s) & \vee & (\neg P_{F_1}(s) \wedge \neg P_{F_2}(s))) \\
\forall s & (\neg P_{F_1 \vee F_2}(s) & \vee & P_{F_1}(s) \vee P_{F_2}(s)) \\[0.5em]
\forall s & (P_{\Box F'}(s) & \vee & (R(s, f_i(s)) \wedge \neg P_{F'}(f_i(s)))) \\
\forall s \forall s' & (\neg P_{\Box F'}(s) & \vee & \neg R(s, s') \vee P_{F'}(s')) \\[0.5em]
\forall s \forall s' & (P_{\Diamond F'}(s) & \vee & \neg R(s, s') \vee \neg P_{F'}(s'))) \\
\forall s & (\neg P_{\Diamond F'}(s) & \vee & (R(s, f_j(s)) \wedge P_{F'}(f_j(s)))) \\[0.5em]
 & \multicolumn{3}{l}{\text{index formulae range over all subformulae of } F.}
\end{array}}_{\text{Rename}(F)}$$

Translation to CNF

$$
\underbrace{
\begin{array}{lrcl}
 & \neg P_F(c) & & \\[1ex]
\forall s & (P_{\neg F'}(s) & \vee & P_{F'}(s)) \\
\forall s & (\neg P_{\neg F'}(s) & \vee & \neg P_{F'}(s)) \\[1ex]
\forall s & (P_{F_1 \wedge F_2}(s) & \vee & \neg P_{F_1}(s) \vee \neg P_{F_2}(s)) \\
\forall s & (\neg P_{F_1 \wedge F_2}(s) & \vee & P_{F_1}(s)) \\
\forall s & (\neg P_{F_1 \wedge F_2}(s) & \vee & P_{F_2}(s)) \\[1ex]
\forall s & (P_{F_1 \vee F_2}(s) & \vee & \neg P_{F_1}(s)) \\
\forall s & (P_{F_1 \vee F_2}(s) & \vee & \neg P_{F_2}(s))) \\
\forall s & (\neg P_{F_1 \vee F_2}(s) & \vee & P_{F_1}(s) \vee P_{F_2}(s)) \\[1ex]
\forall s & (P_{\Box F'}(s) & \vee & R(s, f_i(s)) \\
\forall s & (P_{\Box F'}(s) & \vee & \neg P_{F'}(f_i(s)))) \\
\forall s \forall s' & (\neg P_{\Box F'}(s) & \vee & \neg R(s, s') \vee P_{F'}(s')) \\[1ex]
\forall s \forall s' & (P_{\Diamond F'}(s) & \vee & \neg R(s, s') \vee \neg P_{F'}(s'))) \\
\forall s & (\neg P_{\Diamond F'}(s) & \vee & R(s, f_j(s)) \\[1ex]
\forall s & (\neg P_{\Diamond F'}(s) & \vee & P_{F'}(f_j(s)))) \\
 & \multicolumn{3}{c}{\text{index formulae range over all subformulae of } F.}
\end{array}
}_{\text{Rename}(F)}
$$

Ordered resolution as a decision procedure

Let $\Sigma = (\Omega, \Pi)$, where $\Omega = \{c_1/0, \ldots, c_k/0, f_1/1, \ldots, f_l/1\}$, and $\Pi = \{p_1/1, \ldots, p_n/1, R/2\}$. Let X be a set of variables.

We define an ordering and a selection function as follows.

Ordered resolution as a decision procedure

Ordering:

Given:

- $\succ$ ordering which is total and well founded on ground terms and for all terms u, t, if t occurs as a subterm in u then $u \succ t$.
- $\succ_P$ total order on the predicate symbols s.t. $R \succ_P p_i$ for every i.

An ordering on literals (also denoted by $\succ$) is defined as follows.

Let c be the complexity measure defined for every ground literal L by $c_L = (\text{max}_L, \text{pred}_L, p_L)$ where:

- max_L is the maximal term occurring in L;
- pred_L is the predicate symbol occurring in L; and
- p_L is 1 if L is negative and 0 if L is positive.

Ordered resolution as a decision procedure

Ordering: (ctd.)

Let $c_L = (\max_L, \mathrm{pred}_L, p_L)$ where:

- $\max_L$ is the maximal term occurring in L;
- pred_L is the predicate symbol occurring in L; and
- p_L is 1 if L is negative and 0 if L is positive.

The complexity measure c induces a well-founded ordering $\succ_c$ on ground literals, defined by $L \succ_c L'$ if and only if $c_L > c_{L'}$ in the lexicographic combination of $\succ$, $\succ_P$, and $>$ (where $1 > 0$).

Let $\succ$ be a total and well-founded extension of $\succ_c$.

Example: Assume $R \succ_P P_1 \succ_P P_2$ and $d \succ c$

L:	$\neg P_1(f(f(d))) \succ P_1(f(f(d))) \succ \neg P_2(f(f(d))) \succ R(c, f(d)) \succ \neg R(c, d) \succ R(c, c)$ because
c_L:	$(f(f(d)), P_1, 1) > (f(f(d)), P_1, 0) > (f(f(d)), P_2, 1) > (f(d), R, 0) > (d, R, 1) > (c, R, 0)$

Ordered resolution as a decision procedure

Selection function:

Let S be the selection function that selects all occurrences of negative literals starting with the predicate R.

Ordered resolution as a decision procedure

Notation: If $t, t_1, \dots, t_n$ are terms, we use the following notations.

- Any clause clause of form $(\neg)p_{i_1}(t) \vee \cdots \vee (\neg)p_{i_k}(t)$ is of type $\mathcal{P}(t)$
- Any clause clause of form $C_1 \vee \cdots \vee C_n$, where C_i is of type $\mathcal{P}(t_i)$ is of type $\mathcal{P}(t_1, \dots, t_n)$.

Ordered resolution as a decision procedure

Notation: If $t, t_1, \ldots, t_n$ are terms, we use the following notations.

- Any clause clause of form $(\neg)p_{i_1}(t) \vee \cdots \vee (\neg)p_{i_k}(t)$ is of type $\mathcal{P}(t)$
- Any clause clause of form $C_1 \vee \cdots \vee C_n$, where C_i is of type $\mathcal{P}(t_i)$ is of type $\mathcal{P}(t_1, \ldots, t_n)$.

Consider the following sets of clauses:

$\mathcal{G}$	all clauses of type $\mathcal{P}(c)$ where c is a constant.
$\mathcal{V}$	all clauses of type $\mathcal{P}(x)$ for some variable x.
$\mathcal{V}(f)$	clauses of type $\mathcal{P}(x, f(x))$, for some variable x (where $f/1 \in \Omega$).
$\mathcal{R}^+$	all clauses of the form $\mathcal{P}(x) \vee R(x, f(x))$ for some variable x.
$\mathcal{R}^-$	all clauses of the form $\mathcal{P}(x) \vee \mathcal{P}(y) \vee \neg R(x, y)$ for some variables x, y.

Translation to CNF

$$
\underbrace{
\begin{array}{llclr}
 & \neg P_F(c) & & & \mathcal{P}(c) \\[1ex]
\forall s & (P_{\neg F'}(s) & \vee & P_{F'}(s)) & \mathcal{V}(s) \\
\forall s & (\neg P_{\neg F'}(s) & \vee & \neg P_{F'}(s)) & \mathcal{V}(s) \\[1ex]
\forall s & (P_{F_1 \wedge F_2}(s) & \vee & \neg P_{F_1}(s) \vee \neg P_{F_2}(s)) & \mathcal{V}(s) \\
\forall s & (\neg P_{F_1 \wedge F_2}(s) & \vee & P_{F_1}(s)) & \mathcal{V}(s) \\
\forall s & (\neg P_{F_1 \wedge F_2}(s) & \vee & P_{F_2}(s)) & \mathcal{V}(s) \\[1ex]
\forall s & (P_{F_1 \vee F_2}(s) & \vee & \neg P_{F_1}(s)) & \mathcal{V}(s) \\
\forall s & (P_{F_1 \vee F_2}(s) & \vee & \neg P_{F_2}(s))) & \mathcal{V}(s) \\
\forall s & (\neg P_{F_1 \vee F_2}(s) & \vee & P_{F_1}(s) \vee P_{F_2}(s)) & \mathcal{V}(s) \\[1ex]
\forall s & (P_{\Box F'}(s) & \vee & R(s, f_i(s)) & \mathcal{R}^+ \\
\forall s & (P_{\Box F'}(s) & \vee & \neg P_{F'}(f_i(s)))) & \mathcal{V}(f_i) \\
\forall s \forall s' & (\neg P_{\Box F'}(s) & \vee & \neg R(s, s') \vee P_{F'}(s')) & \mathcal{R}^- \\[1ex]
\forall s \forall s' & (P_{\Diamond F'}(s) & \vee & \neg R(s, s') \vee \neg P_{F'}(s'))) & \mathcal{R}^- \\
\forall s & (\neg P_{\Diamond F'}(s) & \vee & R(s, f_j(s)) & \mathcal{R}^+ \\[1ex]
\forall s & (\neg P_{\Diamond F'}(s) & \vee & P_{F'}(f_j(s)))) & \mathcal{V}(f_i) \\
\multicolumn{5}{c}{\text{index formulae range over all subformulae of } F.}
\end{array}
}_{\text{Rename}(F)}
$$

Ordered resolution as a decision procedure

To be proved:

(1) The set $\mathcal{G} \cup \mathcal{V} \cup \bigcup_f \mathcal{V}(f) \cup \mathcal{R}^+ \cup \mathcal{R}^-$ is finite

(2) $\mathcal{G} \cup \mathcal{V}$ is closed under $\mathrm{Res}_S^{\succ}$.

(3) $\mathcal{G} \cup \mathcal{V} \cup \bigcup_f \mathcal{V}(f)$ is closed under $\mathrm{Res}_S^{\succ}$.

(4) $\mathcal{G} \cup \mathcal{V} \cup \bigcup_f \mathcal{V}(f) \cup \mathcal{R}^+$ is closed under $\mathrm{Res}_S^{\succ}$.

(5) $\mathcal{G} \cup \mathcal{V} \cup \bigcup_f \mathcal{V}(f) \cup \mathcal{R}^+ \cup \mathcal{R}^-$ is closed under $\mathrm{Res}_S^{\succ}$.

Ordered resolution as a decision procedure

We assume that no literals occur several times (eager factoring)

Theorem The set $\mathcal{G} \cup \mathcal{V} \cup \bigcup_f \mathcal{V}(f) \cup \mathcal{R}^+ \cup \mathcal{R}^-$ is finite

Proof: (1) $\mathcal{P}(c)$ contains at most $3^{|\text{Subformulae}(F)|}$ clauses, so if there are m constants then $\mathcal{G}$ contains at most $m3^{|\text{Subformulae}(F)|}$ clauses.

Similarly it can be checked that $\mathcal{V}$ contains (up to remaming of variables) $3^{|\text{Subformulae}(F)}$ clauses.

All literals of clauses in $\mathcal{P}(x, f(x))$ have argument x or $f(x)$. We have therefore $2|\text{Subformulae}(F)|$ literals, hence $3^{2|\text{Subformulae}(F)|}$ clauses.

The number of clauses in $\mathcal{R}^+$ is the same as the number of clauses in $\mathcal{P}(x)$. The number of clauses in $\mathcal{R}^-$ is $|\mathcal{P}(x)|^2$.

Ordered resolution as a decision procedure

Theorem

(2) $\mathcal{G} \cup \mathcal{V}$ is closed under $\mathrm{Res}_S^{\succ}$.

(3) $\mathcal{G} \cup \mathcal{V} \cup \bigcup_f \mathcal{V}(f)$ is closed under $\mathrm{Res}_S^{\succ}$.

(4) $\mathcal{G} \cup \mathcal{V} \cup \bigcup_f \mathcal{V}(f) \cup \mathcal{R}^+$ is closed under $\mathrm{Res}_S^{\succ}$.

(5) $\mathcal{G} \cup \mathcal{V} \cup \bigcup_f \mathcal{V}(f) \cup \mathcal{R}^+ \cup \mathcal{R}^-$ is closed under $\mathrm{Res}_S^{\succ}$.

Proof.

(2) The resolvent of two clauses in $\mathcal{G}$ is in $\mathcal{G}$; the resolvent of two clauses in $\mathcal{V}$ is in $\mathcal{V}$; The resolvent of a clause in $\mathcal{G}$ and one in $\mathcal{V}$ is in $\mathcal{G}$.

(3) No inference is possible between clauses in $\mathcal{G}$ and clauses in $\mathcal{V}(f)$. The resolvent of a clause in $\mathcal{V}$ and one in $\mathcal{V}(f)$ is in $\mathcal{V}$ or in $\mathcal{V}(f)$.

The resolvent of two clauses in $\mathcal{V}(f)$ is in $\mathcal{V}$ or $\mathcal{V}(f)$. No inference is possible between clauses in $\mathcal{V}(f)$ and $\mathcal{V}(g)$ if $f \neq g$ (atoms in maximal literals not unifiable)

Ordered resolution as a decision procedure

Theorem

(2) $\mathcal{G} \cup \mathcal{V}$ is closed under $\mathrm{Res}_S^{\succ}$.

(3) $\mathcal{G} \cup \mathcal{V} \cup \bigcup_f \mathcal{V}(f)$ is closed under $\mathrm{Res}_S^{\succ}$.

(4) $\mathcal{G} \cup \mathcal{V} \cup \bigcup_f \mathcal{V}(f) \cup \mathcal{R}^+$ is closed under $\mathrm{Res}_S^{\succ}$.

(5) $\mathcal{G} \cup \mathcal{V} \cup \bigcup_f \mathcal{V}(f) \cup \mathcal{R}^+ \cup \mathcal{R}^-$ is closed under $\mathrm{Res}_S^{\succ}$.

Proof.

(4) No inferences are possible between two clauses in $\mathcal{R}^+$ (in every clause the maximal literal is a positive R-literal and nothing is selected). No inferences are possible between a clause in $\mathcal{R}^+$ and a clause in $\mathcal{G} \cup \mathcal{V} \cup \bigcup_f \mathcal{V}(f)$.

(5) The resolvent of a clause in $\mathcal{R}^+$ and one in $\mathcal{R}^-$ is a clause in $\mathcal{V} \cup \bigcup_f \mathcal{V}(f)$. No inferences are possible between a clause in $\mathcal{R}^+ \cup \mathcal{R}^-$ and a clause in $\mathcal{G} \cup \mathcal{V} \cup \bigcup_f \mathcal{V}(f)$.

Ordered resolution as a decision procedure

Theorem. $\text{Res}_S^{\succ}$ checks satisfiability of sets of clauses in $\mathcal{G} \cup \mathcal{V} \cup \bigcup_f \mathcal{V}(f) \cup \mathcal{R}^+ \cup \mathcal{R}^-$ in exponential time.

Proof (Idea)

Let N be a set of clauses which is a subset of $\mathcal{G} \cup \mathcal{V} \cup \bigcup_f \mathcal{V}(f) \cup \mathcal{R}^+ \cup \mathcal{R}^-$. All clauses which can be derived from N using $\text{Res}_S^{\succ}$ are in $\mathcal{G} \cup \mathcal{V} \cup \bigcup_f \mathcal{V}(f) \cup \mathcal{R}^+ \cup \mathcal{R}^-$.

The size of this set is exponential in the size of $|\text{Subformulae}(F)|$. This means that at most an exponential number of inferences are needed to generate all clauses in this set.

Until now

Modal logic

Syntax

Semantics

- Kripke models
- global and local entailment; deduction theorem

Correspondence theory

First-order definability

Theorem proving in modal logics

Decidability

Now: Description logics

Description Logics

subfield of Knowledge Representation which is a subfield of AI.

- **Description**– comes from concept description (formal expression which determines a set of individuals with common properties)
- **Logics** – comes from the fact that the semantics of concept description can be defined using logic (a fragment of first-order logic)

Why description logics?

Examples of concepts

teaching assistant, undergraduate, professor

Examples of properties

Every teaching assistant is either not an undergraduated or a professor.

Why description logics?

Examples of concepts

teaching assistant, undergraduate, professor

Examples of properties

Every teaching assistant is either not an undergraduated or a professor.

Formal description in first-order logic

Unary predicates: $\texttt{Teaching-Assistant}$, $\texttt{Undergrad}$, $\texttt{Professor}$

$\forall x \quad \texttt{Teaching-Assistant}(x) \rightarrow \neg\texttt{Undergrad}(x) \vee \texttt{Professor}(x)$

Why description logics?

Examples of concepts

teaching assistant, undergraduate, professor

Examples of properties

Every teaching assistant is either not an undergraduated or a professor.

Formal description in first-order logic

Unary predicates: $\texttt{Teaching-Assistant}, \texttt{Undergrad}, \texttt{Professor}$

$\forall x \quad \texttt{Teaching-Assistant}(x) \rightarrow \neg\texttt{Undergrad}(x) \vee \texttt{Professor}(x)$

More concise description

Concept names: $\texttt{Teaching-Assistant}, \texttt{Undergrad}, \texttt{Professor}$

$\texttt{Teaching-Assistant} \sqsubseteq \neg\texttt{Undergrad} \sqcup \texttt{Professor}$

Why description logics?

If predicate logic is directly used without some kind of restriction, then

- the structure of the knowledge/information is lost;
- the expressive power is too high for having good computational properties and efficient procedures.

Example

$\texttt{Teaching-Assistant} \sqsubseteq \neg\texttt{Undergrad} \sqcup \texttt{Professor}$

$\forall x \quad \texttt{Teaching-Assistant}(x) \rightarrow \neg\texttt{Undergrad}(x) \vee \texttt{Professor}(x)$

A necessary condition in order to be a teaching assistant is to be either not undergraduated or a professor.

Example

$\texttt{Teaching-Assistant} \sqsubseteq \neg\texttt{Undergrad} \sqcup \texttt{Professor}$

$\forall x \quad \texttt{Teaching-Assistant}(x) \rightarrow \neg\texttt{Undergrad}(x) \vee \texttt{Professor}(x)$

A necessary condition in order to be a teaching assistant is to be either not undergraduated or a professor.

When the left-hand side is an atomic concept, the "$\sqsubseteq$" symbol introduces a *primitive definition* – giving only necessary conditions.

$\texttt{Teaching-Assistant} \doteq \neg\texttt{Undergrad} \sqcup \texttt{Professor}$

$\forall x \quad \texttt{Teaching-Assistant}(x) \leftrightarrow \neg\texttt{Undergrad}(x) \vee \texttt{Professor}(x)$

The "$\doteq$" symbol introduces a real definition – with necessary and sufficient conditions.
In general, we can have complex concept expressions at the left-hand side as well.

The description logic ALC: Syntax

Concepts:

- primitive concepts N_C
- complex concepts (built using constructors $\neg, \sqcap, \sqcup, \exists R, \forall R, \top, \bot$)

Roles: N_R

The description logic ALC: Syntax

Concepts:
- primitive concepts N_C
- complex concepts (built using constructors $\neg, \sqcap, \sqcup, \exists R, \forall R, \top, \bot$)

Roles: N_R

Concepts:

$$
\begin{array}{lll}
C := & \top & \\
 & |\bot & \\
 & |A & \text{primitive concept} \\
 & |C_1 \sqcap C_2 & \\
 & |C_2 \sqcup C_2 & \\
 & |\neg C & \\
 & |\forall R.C & \\
 & |\exists R.C &
\end{array}
$$

The description logic ALC: Semantics

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

- $C \in N_C \mapsto C^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
- $R \in N_R \mapsto R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$

We can also interpret "individuals" (as elements of $\Delta^{\mathcal{I}}$).

The description logic ALC

Syntax	Semantics	Name
A	$A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$	primitive concept
R	$R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$	primitive role
$\top$	$\Delta^{\mathcal{I}}$	top
$\bot$	$\emptyset$	bottom
$\neg C$	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$	complement
$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$	conjunction
$C \sqcup D$	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$	disjunction
$\forall R.C$	$\{x \mid \forall y \ \ R^{\mathcal{I}}(x, y) \rightarrow y \in C^{\mathcal{I}}\}$	universal quantification (universal role restriction)
$\exists R.C$	$\{x \mid \exists y \ \ R^{\mathcal{I}}(x, y) \ \wedge y \in C^{\mathcal{I}}\}$	existential quantification (existential role restriction)

The description logic ALC: Semantics

- **Conjunction** is interpreted as *intersection* of sets of individuals.
- **Disjunction** is interpreted as *union* of sets of individuals.
- **Negation** is interpreted as *complement* of sets of individuals.

For every interpretation $\mathcal{I}$:

- $(\neg(C \sqcap D))^{\mathcal{I}} = (\neg C \sqcup \neg D)^{\mathcal{I}}$
- $(\neg(C \sqcup D))^{\mathcal{I}} = (\neg C \sqcap \neg D)^{\mathcal{I}}$
- $(\neg(\forall R.C))^{\mathcal{I}} = (\exists R.\neg C)^{\mathcal{I}}$
- $(\neg(\exists R.C))^{\mathcal{I}} = (\forall R.\neg C)^{\mathcal{I}}$

Knowledge Bases

- **Terminological Axioms (TBox):** $C \sqsubseteq D$, $C \doteq D$
  - $\mathtt{Student} \doteq \mathtt{Person} \sqcap \exists\mathtt{NAME.String} \sqcap$
    $\exists\mathtt{ADDRESS.String} \sqcap$
    $\exists\mathtt{ENROLLED.Course}$
  - $\mathtt{Student} \sqsubseteq \exists\mathtt{ENROLLED.Course}$
  - $\exists\mathtt{TEACHES.Course} \sqsubseteq \neg\mathtt{Undergrad} \sqcup \mathtt{Professor}$

- **Membership statements (ABox):** $C(a)$, $R(a, b)$
  - $\mathtt{Student}(\mathtt{john})$
  - $\mathtt{ENROLLED}(\mathtt{john}, \mathtt{cs415})$
  - $(\mathtt{Student} \sqcup \mathtt{Professor})(\mathtt{paul})$

Semantics

We consider the descriptive semantics, based on classical logics.

- An interpretation $\mathcal{I}$ *satisfies* the statement $C \sqsubseteq D$ if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$.
- An interpretation $\mathcal{I}$ *satisfies* the statement $C \doteq D$ if $C^{\mathcal{I}} = D^{\mathcal{I}}$.

An interpretation $\mathcal{I}$ is a *model* for a TBox $\mathcal{T}$ if $\mathcal{I}$ satisfies all the statements in $\mathcal{T}$.

ABox

A set $\mathcal{A}$ of assertions (membership or relationship statements) is called an ABox.

If $\mathcal{I} = (D^{\mathcal{I}}, \cdot_{\mathcal{I}})$ is an interpretation,

- $C(a)$ is satisfied by $\mathcal{I}$ if $a^{\mathcal{I}} \in C^{\mathcal{I}}$.
- $R(a, b)$ is satisfied by $\mathcal{I}$ if $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$.

An interpretation $\mathcal{I}$ is said to be a *model* of the ABox $\mathcal{A}$ if every assertion of $\mathcal{A}$ is satisfied by $\mathcal{I}$.

The ABox $\mathcal{A}$ is said to be *satisfiable* if it admits a model.

Semantics

An interpretation $\mathcal{I} = (D^{\mathcal{I}}, \cdot_{\mathcal{I}})$ is said to be a *model* of a knowledge base $(\mathcal{T}, \mathcal{A})$ if every axiom of the knowledge base is satisfied by $\mathcal{I}$.

A knowledge base $(\mathcal{T}, \mathcal{A})$ is said to be *satisfiable* if it admits a model.

Logical Implication

$(\mathcal{T}, \mathcal{A}) \models \varphi$ if every model of $(\mathcal{T}, \mathcal{A})$ is a model of φ

Example 1:

- TBox: $\mathcal{T}$
  - $$\begin{aligned} \texttt{Student} \doteq \texttt{Person} \sqcap {} & \exists \texttt{NAME}.\texttt{String} \sqcap \\ & \exists \texttt{ADDRESS}.\texttt{String} \sqcap \\ & \exists \texttt{ENROLLED}.\texttt{Course} \end{aligned}$$
  - $\texttt{Student} \sqsubseteq \exists \texttt{ENROLLED}.\texttt{Course}$
  - $\exists \texttt{TEACHES}.\texttt{Course} \sqsubseteq \neg \texttt{Undergrad} \sqcup \texttt{Professor}$
- ABox: $\mathcal{A} = \emptyset$

$$(\mathcal{T}, \mathcal{A}) \overset{?}{\models} \texttt{Student} \sqsubseteq \texttt{Person}$$

Logical Implication

$(\mathcal{T}, \mathcal{A}) \models \varphi$ if every model of $(\mathcal{T}, \mathcal{A})$ is a model of φ

Example 2:

TBox: $\mathcal{T}$

$\exists \texttt{TEACHES}.\texttt{Course} \sqsubseteq \neg \texttt{Undergrad} \sqcup \texttt{Professor}$

ABox: $\mathcal{A}$

$\texttt{TEACHES}(\texttt{john}, \texttt{cs415}), \texttt{Course}(\texttt{cs415}),$
$\texttt{Undergrad}(\texttt{john})$

$(\mathcal{T}, \mathcal{A}) \models \texttt{Professor}(\texttt{john})$

Logical Implication

TBox: $\mathcal{T}$

$\exists\texttt{TEACHES}.\texttt{Course} \sqsubseteq$

$\neg\texttt{Undergrad} \sqcup \texttt{Professor}$

ABox: $\mathcal{A}$

$\texttt{TEACHES}(\texttt{john}, \texttt{cs415}), \texttt{Course}(\texttt{cs415}),$

$\texttt{Undergrad}(\texttt{john})$

$$(\mathcal{T}, \mathcal{A}) \overset{?}{\models} \texttt{Professor}(\texttt{john})$$

$$(\mathcal{T}, \mathcal{A}) \overset{?}{\models} \neg\texttt{Professor}(\texttt{john})$$

Reasoning Problems

- **Concept Satisfiability**

  $(\mathcal{T}, \mathcal{A}) \not\models C \equiv \bot$ Example: $\texttt{Student} \sqcap \neg\texttt{Person}$

  the problem of checking whether C is satisfiable w.r.t. Σ, i.e. whether there exists a model $\mathcal{I}$ of Σ such that $C^{\mathcal{I}} \neq \emptyset$

- **Subsumption**

  $(\mathcal{T}, \mathcal{A}) \models C \sqsubseteq D$ Example: $\texttt{Student} \sqsubseteq \texttt{Person}$

  the problem of checking whether C is subsumed by D w.r.t. Σ, i.e. whether $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ in every model $\mathcal{I}$ of $(\mathcal{T}, \mathcal{A})$

- **Satisfiability**

  $(\mathcal{T}, \mathcal{A}) \not\models \text{false}$

  the problem of checking whether $(\mathcal{T}, \mathcal{A})$ is satisfiable, i.e. whether it has a model

- **Instance Checking**

  $(\mathcal{T}, \mathcal{A}) \models C(a)$ Example: $\texttt{Professor}(\texttt{john})$

  the problem of checking whether the assertion $C(a)$ is satisfied in every model of $(\mathcal{T}, \mathcal{A})$

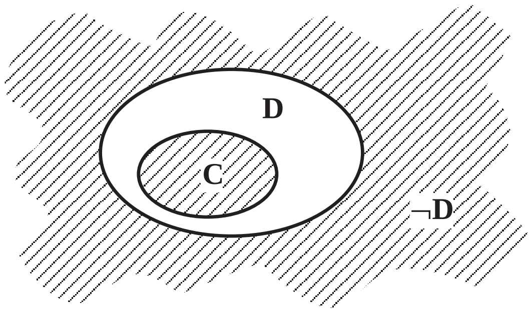
Reduction to concept satisfiability

- **Concept Satisfiability**

  $(\mathcal{T}, \mathcal{A}) \not\models C \equiv \bot \quad \leftrightarrow$
  $\mathcal{T} \cup \mathcal{A} \cup \{C(x)\}$ has a model

- **Subsumption**

  $(\mathcal{T}, \mathcal{A}) \models C \sqsubseteq D \quad \leftrightarrow$
  $(\mathcal{T}, \mathcal{A}) \models C \sqcap \neg D \equiv \bot \quad \leftrightarrow$
  $(\mathcal{T}, \mathcal{A}) \cup \{(C \sqcap \neg D)(x)\}$ has no models



- **Instance Checking**

  $(\mathcal{T}, \mathcal{A}) \models C(a) \quad \leftrightarrow$
  $(\mathcal{T}, \mathcal{A}) \cup \{\neg C(a)\}$ has no models

Other reasoning problems

Classification

- Given a concept C and a TBox T, for all concepts D of T determine whether D subsumes C, or D is subsumed by C.
- Intuitively, this amounts to finding the "right place" for C in the taxonomy implicitly present in T.
- *Classification* is the task of inserting new concepts in a taxonomy. It is *sorting* in partial orders.

Goal

- Prove decidability of description logic
- Give efficient decision procedures

Goal

- Prove decidability of description logic
- Give efficient decision procedures

$\mathcal{ALC}$: Express it as a multi-modal logic