Non-classical logics

Lectures 19 and 20: Modal logics (Part 6)

Viorica Sofronie-Stokkermans

sofronie@uni-koblenz.de

Until now

- History and Motivation
- Propositional modal logic

  Syntax/Semantics

  Correspondence theory; First-order definability

  Proof Systems (Inference system; Tableau calculi; Resolution)

  Decidability
- Description logics

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):** $C \sqsubseteq D$, $C \doteq D$
- **Membership statements (ABox):** $C(a)$, $R(a, b)$

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.

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**

  $(\mathcal{T}, \mathcal{A}) \not\models C \equiv \bot \quad \leftrightarrow$

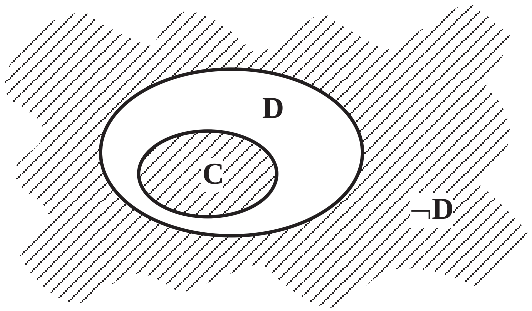
  $\mathcal{T} \cup \mathcal{A} \cup \{C(x)\}$ has a model

- **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

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

We translate every concept C of $\mathcal{ALC}$ into a formula F_C in a many-modal logic which contains modal operators

$$\Box_R, \Diamond_R \quad \text{for every role } R$$

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

We translate every concept C of $\mathcal{ALC}$ into a formula in a many-modal logic which contains modal operators

$$\Box_R, \Diamond_R \quad \text{for every role } R$$

In the translation we replace every primitive concept symbol with a propositional variable.

$$C \quad \mapsto \quad F_C := C \quad \text{if } C \text{ is a primitive concept}$$

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

We translate every concept C of $\mathcal{ALC}$ into a formula in a many-modal logic which contains modal operators

$$\Box_R, \Diamond_R \qquad \text{for every role } R$$

In the translation we replace every primitive concept symbol with a propositional variable.

$$
\begin{array}{lcll}
C & \mapsto & F_C := C & \text{if } C \text{ is a primitive concept} \\
C_1 \sqcap C_2 & \mapsto & F_{C_1 \sqcap C_2} := F_{C_1} \wedge F_{C_2} & \\
C_1 \sqcup C_2 & \mapsto & F_{C_1 \sqcup C_2} := F_{C_1} \vee F_{C_2} & \\
\neg C & \mapsto & F_{\neg C} := \neg F_C & \\
\forall R.C & \mapsto & F_{\forall R.C} := \Box_R F_C & \\
\exists R.C & \mapsto & F_{\exists R.C} := \Diamond_R F_C &
\end{array}
$$

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

An interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ where

$C^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$

$R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$

clearly corresponds to a (multi-modal) Kripke structure
$\mathcal{K} = (S, \{\rho_R\}_{R \in N_R}, I)$ where

- $S = \Delta^{\mathcal{I}}$
- $\rho_R = R^{\mathcal{I}}$
- $I : \Pi \times S \to \{0, 1\}$ (where $\Pi = N_C$) is defined by:
  $I(C, x) = 1$ iff $x \in C^{\mathcal{I}}$

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

Lemma. For every $\mathcal{ALC}$ concept C and every interpretation $\mathcal{I}$ we have:

$$C^{\mathcal{I}} = \{d \in \Delta^{\mathcal{I}} \mid (\mathcal{K}, d) \models F_C\}.$$

Proof: Structural induction

If $C \in N_C$ the result follows from the way the valuation of $\mathcal{K}$ is defined.

For the induction step we here only consider the case $C = \forall R.C_1$
Induction hypothesis (IH): property holds for C_1.

$$\begin{array}{ll}
\{d \in \Delta^{\mathcal{I}} \mid (\mathcal{K}, d) \models F_C\} = \{d \in \Delta^{\mathcal{I}} \mid (\mathcal{K}, d) \models F_{\forall R.C_1}\} & = \\
\{d \in \Delta^{\mathcal{I}} \mid (\mathcal{K}, d) \models \Box_R F_{C_1}\} & = \\
\{d \in \Delta^{\mathcal{I}} \mid \text{ for all } e \text{ with } R(d, e) \text{ we have } (\mathcal{K}, e) \models F_{C_1}\} & \stackrel{IH}{=} \\
\{d \in \Delta^{\mathcal{I}} \mid \text{ for all } e \text{ with } R(d, e) \text{ we have } e \in C_1^{\mathcal{I}}\} & = (\forall R.C_1)^{\mathcal{I}} = C^{\mathcal{I}}
\end{array}$$

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

Lemma There exists an interpretation $\mathcal{I}$ and a $d \in \Delta^{\mathcal{I}}$ such that $d \in C^{\mathcal{I}}$ iff F_C is satisfiable in the multi-modal logic.

Proof Immediate consequence of the previous lemma.

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

Lemma $C_1 \sqsubseteq C_2$ iff $F_{C_1 \sqcap \neg C_2}$ is unsatisfiable in the multi-modal logic.

Proof. $C_1 \sqsubseteq C_2$ iff for all $\mathcal{I}$ and all $d \in \Delta^{\mathcal{I}}$ we have: $d \notin (C_1 \sqcap \neg C_2)^{\mathcal{I}}$

From the first lemma, this happens iff $(\mathcal{K}, d) \not\models F_{C_1} \wedge \neg F_{C_2}$ for all $\mathcal{I}$ and all $d \in \Delta^{\mathcal{I}}$.

This is the same as saying that $F_{C_1 \sqcap \neg C_2}$ is unsatisfiable.

Reasoning procedures

- Terminating, efficient and complete algorithms for deciding **satisfiability** – and all the other reasoning services – are available.
- Algorithms are based on tableaux-calculi techniques or resolution.

Description logics

Two directions of research:

- Extensions in order to increase expressivity

- Restrict language in order to identify “tractable” description logics

Description logics

Two directions of research:

- Extensions in order to increase expressivity

  SHIQ

- Restrict language in order to identify "tractable" description logics

  $\mathcal{EL}$

Some extensions of ALC

SHIQ:

Syntax:

N_C primitive concept symbols

N_R^0 set of atomic role symbols

$N_t^0 \subseteq N_R^0$ set of transitive role symbols

The set N_R of role symbols contains all atomic roles and for every role $R \in N_R^0$ also its inverse role R^-.

Some extensions of ALC

SHIQ:

Role hierarchy:

A role hierarchy is a finite set $\mathcal{H}$ of formulae of the form

$$R_1 \sqsubseteq R_2$$

for $R_1, R_2 \in N_R$.

All following definitions assume that a role hierarchy is given (and fixed)

SHIQ concept descriptions: Syntax

$$
\begin{array}{lcll}
C & := & A & \text{if } A \text{ is a primitive concept} \\
 & & |\top & \\
 & & |\neg C & \\
 & & |C_1 \sqcap C_2 & \\
 & & |C_2 \sqcup C_2 & \\
 & & |\exists R.C & \\
 & & |\forall R.C & \\
 & & | \leq nR.C & \text{where } n \in \mathbb{N}, R \text{ simple role} \\
 & & | \geq nR.C & \text{where } n \in \mathbb{N}, R \text{ simple role}
\end{array}
$$

R is a simple role if $R \notin N_t^0$ and R does not contain any transitive sub-role.

SHIQ concept descriptions: Syntax

$$
\begin{array}{lll}
C & := & A \qquad\qquad\qquad \text{if } A \text{ is a primitive concept} \\
 & & |\top \\
 & & |\neg C \\
 & & |C_1 \sqcap C_2 \\
 & & |C_2 \sqcup C_2 \\
 & & |\exists R.C \\
 & & |\forall R.C \\
 & & | \leq nR.C \qquad \text{where } n \in \mathbb{N}, R \text{ simple role} \\
 & & | \geq nR.C \qquad \text{where } n \in \mathbb{N}, R \text{ simple role}
\end{array}
$$

R is a simple role if $R \not\in N_t^0$ and R does not contain any transitive sub-role.

Abbreviations: $\geq nR := \geq nR.\top \quad \leq nR := \geq nR.\top$

Cardinality Restriction

Role quantification cannot express that a woman has *at least 3* (or *at most 5*) children.

Cardinality restrictions can express conditions on the number of fillers:

- $\mathtt{Busy{-}Woman} \doteq \mathtt{Woman} \sqcap (\geq 3\mathtt{CHILD})$
- $\mathtt{Woman{-}with{-}at{-}most5children} \doteq \mathtt{Woman} \sqcap (\leq 5\mathtt{CHILD})$

$$\boxed{(\geq 1 R) \Longleftrightarrow (\exists R)}$$

Interpretations for SHIQ

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

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

such that:

- for all $R \in N_t^0$, $R^{\mathcal{I}}$ is a transitive relation
- for all $R \in N_R^0$, $(R^{-1})^{\mathcal{I}}$ is the inverse of $R^{\mathcal{I}}$
- for all $R_1 \sqsubseteq R_2 \in \mathcal{H}$ we have $R_1^{\mathcal{I}} \subseteq R_2^{\mathcal{I}}$

SHIQ constructors: Semantics

Constructor	Syntax	Semantics
concept name	A	$A^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
top	$\top$	$D^{\mathcal{I}}$
bottom	$\bot$	$\emptyset$
conjunction	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
disjunction	$C \sqcup D$	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$
negation	$\neg C$	$D^{\mathcal{I}} \setminus C^{\mathcal{I}}$
universal	$\forall R.C$	$\{x \mid \forall y(R^{\mathcal{I}}(x,y) \rightarrow y \in C^{\mathcal{I}})\}$
existential	$\exists R.C$	$\{x \mid \exists y(R^{\mathcal{I}}(x,y) \wedge y \in C^{\mathcal{I}}\}$
cardinality	$\geq nR$	$\{x \mid \#\{y \mid R^{\mathcal{I}}(x,y)\} \geq n\}$
	$\leq nR$	$\{x \mid \#\{y \mid R^{\mathcal{I}}(x,y)\} \leq n\}$
qual. cardinality	$\geq nR.C$	$\{x \mid \#\{y \mid R^{\mathcal{I}}(x,y) \wedge y \in C^{\mathcal{I}}\} \geq n\}$
	$\leq nR.C$	$\{x \mid \#\{y \mid R^{\mathcal{I}}(x,y) \wedge y \in C^{\mathcal{I}}\} \leq n\}$

Decidability

Theorem. The satisfiability and subsumption problem for $\mathcal{SHIQ}$ are decidable

Proof: cf. Horrocks et al.

Undecidability

Theorem. If in the definition of SHIQ we do not impose the restriction about simple roles, the satisfiability problem becomes undecidable

(even if we only allow for cardinality restrictions of the form $\leq nR.\top$ and $\geq nR.\top$).

Proof: cf. Horrocks et al.

Reasoning procedures

- For decidable description logic it is important to have efficient reasoning procedures which are sound, complete and termination.
- Literature: tableau calculi

Goals:

- Completeness is important for the usability of description logics in real applications.
- Efficiency: Algorithms need to be efficient for both average and real knowledge bases, even if the problem in the corresponding logic is in PSPACE or EXPTIME.

A tractable DL

Tractable description logic: $\mathcal{EL}, \mathcal{EL}^+$ and extensions [Baader'03–]
used e.g. in medical ontologies (SNOMED)

$\mathcal{EL}$: Generalities

Concepts: • primitive concepts N_C

• complex concepts (built using concept constructors $\sqcap$, $\exists r$)

Roles: N_R

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

• $C \in N_C \mapsto C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$

• $r \in N_R \mapsto r^{\mathcal{I}} \subseteq D^{\mathcal{I}} \times D^{\mathcal{I}}$

Constructor name	Syntax	Semantics
conjunction	$C_1 \sqcap C_2$	$C_1^{\mathcal{I}} \cap C_2^{\mathcal{I}}$
existential restriction	$\exists r.C$	$\{x \mid \exists y((x, y) \in r^{\mathcal{I}} \text{ and } y \in C^{\mathcal{I}})\}$

$\mathcal{EL}$: Generalities

Concepts:
- primitive concepts N_C
- complex concepts (built using concept constructors $\sqcap, \exists r$)

Roles: N_R

Interpretations: $\mathcal{I} = (D^{\mathcal{I}}, \cdot^{\mathcal{I}})$
- $C \in N_C \mapsto C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
- $r \in N_R \mapsto r^{\mathcal{I}} \subseteq D^{\mathcal{I}} \times D^{\mathcal{I}}$

Problem:

Given: TBox (set $\mathcal{T}$ of concept inclusions $C_i \sqsubseteq D_i$)

concepts C, D

Task: test whether $C \sqsubseteq_{\mathcal{T}} D$, i.e. whether for all $\mathcal{I} = (D^{\mathcal{I}}, \cdot^{\mathcal{I}})$

if $C_i^{\mathcal{I}} \subseteq D_i^{\mathcal{I}} \;\; \forall C_i \sqsubseteq D_i \in \mathcal{T}$ then $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$

$\mathcal{EL}$: Example

Primitive concepts:	protein, process, substance
Roles:	catalyzes, produces
Terminology:	enzyme $=$ protein $\sqcap$ $\exists$catalyzes.reaction
(TBox)	catalyzer $=$ $\exists$catalyzes.process
	reaction $=$ process $\sqcap$ $\exists$produces.substance
Query:	enzyme $\sqsubseteq$ catalyzer?

$\mathcal{EL}^+$: generalities

Concepts:
- primitive concepts N_C
- complex concepts (built using concept constructors $\sqcap, \exists r$)

Roles: N_R

Interpretations: $\mathcal{I} = (D^\mathcal{I}, \cdot^\mathcal{I})$
- $C \in N_C \mapsto C^\mathcal{I} \subseteq D^\mathcal{I}$
- $r \in N_R \mapsto r^\mathcal{I} \subseteq D^\mathcal{I} \times D^\mathcal{I}$

Problem:

Given: CBox $\mathcal{C} = (\mathcal{T}, RI)$, where $\mathcal{T}$ set of concept inclusions $C_i \sqsubseteq D_i$;
RI set of role inclusions $r \circ s \sqsubseteq t$ or $r \sqsubseteq t$
concepts C, D

Task: test whether $C \sqsubseteq_\mathcal{C} D$, i.e. whether for all $\mathcal{I} = (D^\mathcal{I}, \cdot^\mathcal{I})$
if $C_i^\mathcal{I} \subseteq D_i^\mathcal{I} \quad \forall C_i \sqsubseteq D_i \in \mathcal{T}$ and
$r^\mathcal{I} \circ s^\mathcal{I} \subseteq t^\mathcal{I} \quad \forall r \circ s \sqsubseteq t \in RI$ then $C^\mathcal{I} \subseteq D^\mathcal{I}$

$\mathcal{EL}^+$: Example

Primitive concepts:	protein, process, substance
Roles:	catalyzes, produces, helps-producing
Terminology:	enzyme $=$ protein $\sqcap$ $\exists$catalyzes.reaction
(TBox)	reaction $=$ process $\sqcap$ $\exists$produces.substance
Role inclusions:	catalyzes $\circ$ produces $\sqsubseteq$ helps-producing
Query:	enzyme $\sqsubseteq$ protein $\sqcap$ $\exists$helps-producing.substance ?

Complexity

T-Box subsumption for $\mathcal{EL}$ decidable in PTIME

C-Box subsumption for $\mathcal{EL}^+$ decidable in PTIME

Methods:

Reductions to checking satisfiability of clauses in propositional logic.

$\mathcal{EL}$: Hierarchical reasoning

Primitive concepts:	protein, process, substance
Roles:	catalyzes, produces
Terminology:	enzyme $=$ protein $\sqcap$ $\exists$catalyzes.reaction
(TBox)	catalyzer $=$ $\exists$catalyzes.process
	reaction $=$ process $\sqcap$ $\exists$produces.substance
Query:	enzyme $\sqsubseteq$ catalyzer?

$$
\begin{aligned}
\text{SLat} \cup \text{Mon} \models\; & \text{enzyme} = \text{protein} \sqcap \text{catalyzes-some}(\text{reaction}) \quad \wedge \\
& \text{catalyzer} = \text{catalyze-some}(\text{process}) \quad \wedge \\
& \text{reaction} = \text{process} \sqcap \text{produces-some}(\text{substance}) \\
& \Rightarrow \text{enzyme} \sqsubseteq \text{catalyzer}
\end{aligned}
$$

$$
\begin{aligned}
\text{Mon}: \; & \forall C, D (C \sqsubseteq D \rightarrow \text{catalyze-some}(C) \sqsubseteq \text{catalyze-some}(D)) \\
& \forall C, D (C \sqsubseteq D \rightarrow \text{produces-some}(C) \sqsubseteq \text{produces-some}(D))
\end{aligned}
$$

$\mathcal{EL}$: Hierarchical reasoning

$$\text{SLat} \cup \text{Mon} \wedge \underbrace{\begin{array}{l} \text{enzyme} = \text{protein} \sqcap \text{catalyzes-some}(\text{reaction}) \wedge \\ \text{catalyzer} = \text{catalyze-some}(\text{process}) \wedge \\ \text{reaction} = \text{process} \sqcap \text{produces-some}(\text{substance}) \wedge \\ \text{enzyme} \not\sqsubseteq \text{catalyzer} \end{array}}_{G} \models \perp$$

$G \wedge \text{Mon}$
enzyme $=$ protein $\sqcap$ catalyzes-some(reaction) $\wedge$
catalyzer $=$ catalyze-some(process) $\wedge$
reaction $=$ process $\sqcap$ produces-some(substance) $\wedge$
enzyme $\not\sqsubseteq$ catalyzer
$\forall C, D(C \sqsubseteq D \rightarrow \text{catalyze-some}(C) \sqsubseteq \text{catalyze-some}(D))$
$\forall C, D(C \sqsubseteq D \rightarrow \text{produces-some}(C) \sqsubseteq \text{produces-some}(D))$

$\mathcal{EL}$: Hierarchical reasoning

$$\text{SLat} \cup \text{Mon} \wedge \underbrace{\begin{array}{l} \text{enzyme} = \text{protein} \sqcap \text{catalyzes-some}(\text{reaction}) \wedge \\ \text{catalyzer} = \text{catalyze-some}(\text{process}) \wedge \\ \text{reaction} = \text{process} \sqcap \text{produces-some}(\text{substance}) \wedge \\ \text{enzyme} \not\leq \text{catalyzer} \end{array}}_{G} \models \perp$$

Solution 1: Use $DPLL(\text{SLat} + UIF)$

$G \wedge \text{Mon}[G]$
enzyme $=$ protein $\sqcap$ catalyzes-some(reaction)
catalyzer $=$ catalyzes-some(process)
reaction $=$ process $\sqcap$ produces-some(substance)
enzyme $\not\leq$ catalyzer
reaction $\rhd$ process $\rightarrow$ catalyzes-some(reaction) $\rhd$ catalyzes-some(process), $\rhd \in \{\leq, \geq, =\}$

$\mathcal{EL}$: Hierarchical reasoning

$$\text{SLat} \cup \text{Mon} \wedge \underbrace{\begin{array}{l} \text{enzyme} = \text{protein} \sqcap \text{catalyzes-some}(\text{reaction}) \wedge \\ \text{catalyzer} = \text{catalyze-some}(\text{process}) \wedge \\ \text{reaction} = \text{process} \sqcap \text{produces-some}(\text{substance}) \wedge \\ \text{enzyme} \not\leq \text{catalyzer} \end{array}}_{G} \models \perp$$

Solution 2: Hierarchical reasoning

Base theory (SLat)	Extension
enzyme $=$ protein $\sqcap$ c_1	c_1 $=$ catalyzes-some(reaction)
catalyzer $=$ c_2	c_2 $=$ catalyzes-some(process)
reaction $=$ process $\sqcap$ c_3	c_3 $=$ produces-some(substance)
enzyme $\not\leq$ catalyzer	
reaction $\rhd$ process $\rightarrow$ $c_1 \rhd c_2$ $\quad \rhd \in \{\leq, \geq, =\}$	

Test satisfiability using any prover for SLat (e.g. reduction to SAT)

$\mathcal{EL}$: Hierarchical reasoning

Idea in the translation to SAT:

Base theory $\mapsto$	SAT (FOL)
enzyme $=$ protein $\sqcap$ c_1	$\forall x$ enzyme$(x) \leftrightarrow$ protein$(x) \wedge c_1(x)$
catalyzer $=$ c_2	$\forall x$ catalyzer$(x) \leftrightarrow c_2(x)$
reaction $=$ process $\sqcap$ c_3	$\forall x$ reaction$(x) \leftrightarrow$ process$(x) \wedge c_3(x)$
enzyme $\not\sqsubseteq$ catalyzer	enzyme$(c) \wedge \neg$catalyzer(c)
reaction $\sqsubseteq$ process $\rightarrow c_1 \sqsubseteq c_2$	$(\forall x($reaction$(x) \rightarrow$ process$(x))) \rightarrow (\forall x(c_1(x) \rightarrow c_2(x)))$
...	

$$\Downarrow$$

$$(\text{reaction}(d) \rightarrow \text{process}(d)) \rightarrow (\forall x(c_1(x) \rightarrow c_2(x)))$$

$$\Downarrow$$

Clause normal form: no function symbols of arity ≥ 1; Horn except for last class of clauses (a small amount of case distinction $\mapsto$ no increase in compl.)

By Herbrand's theorem the set of clauses is satisfiable iff its set of instances is. Size of instantiated set: polynomial. Satisfiability of Horn clauses: in PTIME.

Dynamic Logic

Motivation

A Simple Programming Language

Logical basis

Typed first-order predicate logic
(Types, variables, terms, formulas, . . .)

Assumption for examples

The signature contains a type Nat and appropriate symbols:

- function symbols $0, s, +, *$
  (terms s(0), s(s(0)), . . . written as 1,2, . . .)
- predicate symbols $\doteq, \leq, <, \geq, >$

NOTE: This is a “convenient assumption” not a definition

Motivation

Programs

- Assignments: $X := t$ $\quad$ X: variable, t:term
- Test: if B then a else b fi

  B: quant.-free formula, a, b: programs
- Loop: while B do a od

  B: quantifier-free formula, a: program
- Composition: $a; b$ $\quad$ a, b programs

WHILE is computationally complete

Motivation

WHILE: Examples

Compute the square of X and store it in Y

$$Y := X * X$$

If X is positive then add one else subtract one

$$\text{if } X > 0 \text{ then } X := X + 1 \text{ else } X := X - 1 \text{ fi}$$

Motivation

WHILE: Example - Square of a Number

Compute the square of X (the complicated way)

Making use of: $n^2 = 1 + 3 + 5 + \cdots + (2 * n - 1)$

```
I := 0;
Y := 0;
while I < X do
Y :=Y +2*I+1;
I := I+1
od
```

Motivation

WHILE: Operational Semantics

Given

A (fixed) first-order structure $\mathcal{A}$ interpreting the function and predicate symbols in the signature

State

$s = (\mathcal{A}, \beta)$ where β is a variable assignment (i.e. function interpreting the variables)

Motivation

State update

$s[e/X] = (\mathcal{A}, \beta[X \mapsto e])$

with $\beta[X \mapsto e](Y) = \begin{cases} e & \text{if } Y = X \\ \beta(Y) & \text{otherwise} \end{cases}$

Motivation

Define the relation $R(\alpha)$ as follows (we write $s[\alpha]s'$ instead of $sR(\alpha)s'$):

- $s[X := t]s'$ iff $s' = s[s(t)/X]$
- $s[\text{if } B \text{ then } \alpha \text{ else } \beta \text{ fi}]s'$ iff $s \models B$ and $s[\alpha]s'$ or $s \models \neg B$ and $s[\beta]s'$.
- $s[\text{while } B \text{ do } \alpha \text{ od}]s'$ iff there are states $s = s_0, \ldots, s_t = s'$ s.t.
  $s_i \models B$ for $0 \leq i \leq t-1$ and $s_t \models \neg B$ and $s_0[\alpha]s_1, s_1[\alpha]s_2, \ldots, s_{t-1}[\alpha]s_t$
- $s[\alpha; \beta]s'$ iff there is a state s'' such that $s[\alpha]s''$ and $s''[\beta]s'$

If α is a deterministic program, $[\alpha]$ is a partial function

Motivation

A Different Approach to WHILE

Programs

- $X := t$ (atomic program)
- $\alpha; \beta$ (sequential composition)
- $\alpha \cup \beta$ (non-deterministic choice)
- α^* (non-deterministic iteration, n times for some $n \geq 0$)
- $F?$ (test)
  remains in initial state if F is true,
  does not terminate if F is false

Motivation

Restriction to deterministic programs

Non-deterministic program constructors may only be used in

if B then α else β fi $\equiv (B?;\alpha) \cup ((\neg B)?;\beta)$

while B do α od $\equiv (B?;\alpha)^*;(\neg B)?$

Motivation

Expressing Program Properties

Logic for expressing properties

Full first-order logic (usually with arithmetic)

Partial correctness assertion (Hoare formula)

$$\{P\}\alpha\{Q\}$$

Meaning:
If α is started in a state satisfying P and terminates, then its final state satisfies Q

Formally:
$\{P\}\alpha\{Q\}$ is valid iff for all states s, s', if $s \models P$ and $s[\alpha]s'$, then $s' \models Q$

Examples

$\{X > 0\} X := X + 1 \{X > 1\}$

$\{\text{even}(X)\} X := X + 2 \{\text{even}(X)\}$
where $\text{even}(X) \equiv \exists Z (X = 2 * Z)$

$\{true\} \alpha_{\text{square}} \{Y = X * X\}$

Examples

$\{X > 0\}X := X + 1\{X > 1\}$

$\{\text{even}(X)\}X := X + 2\{\text{even}(X)\}$
  where $\text{even}(X) \equiv \exists Z(X = 2 * Z)$

$\{\mathit{true}\}\alpha_{\text{square}}\{Y = X * X\}$

Verification: Use annotation of programs with “invariants”

Dynamic Logic

The idea of dynamic logic

- Annotated programs use formulas within programs
- Dynamic Logic uses programs within formulas
- Instead of "assert F" after program segment α, write: $[\alpha]F$

$\mapsto$ multi-modal logic

Dynamic Logic

Dynamic logic is a language for specifying programming languages.

The original work on dynamic logic is by Vaughan Pratt (1976) and by David Harel (1979).

Propositional Dynamic Logic

Propositional dynamic logic (PDL) is a multi-modal logic with structured modalities.

For each program α, there is:
– a box-modality $[\alpha]$ and
– a diamond modality $\langle\alpha\rangle$.

PDL was developed from first-order dynamic logic by Fischer-Ladner (1979) and has become popular recently.

Here we consider regular PDL.

Propositional Dynamic Logic

Syntax

Prog set of programs

$\text{Prog}_0 \subseteq \text{Prog}$: set of atomic programs

Π: set of propositional variables

The set of formulae $\text{Fma}^{PDL}_{\text{Prog},\Pi}$ of (regular) propositional dynamic logic and the set of programs P_0 are defined by simultaneous induction as follows:

PDL: Syntax

Formulae:

$$
\begin{array}{lcll}
F, G, H & ::= & \bot & \text{(falsum)} \\
& \mid & \top & \text{(verum)} \\
& \mid & p & p \in \Pi_0 \text{ (atomic formula)} \\
& \mid & \neg F & \text{(negation)} \\
& \mid & (F \wedge G) & \text{(conjunction)} \\
& \mid & (F \vee G) & \text{(disjunction)} \\
& \mid & (F \rightarrow G) & \text{(implication)} \\
& \mid & (F \leftrightarrow G) & \text{(equivalence)} \\
& \mid & [\alpha]F & \text{if } \alpha \in \text{Prog} \\
& \mid & \langle\alpha\rangle F & \text{if } \alpha \in \text{Prog}
\end{array}
$$

Programs:

$$
\begin{array}{lcll}
\alpha, \beta, \gamma & ::= & \alpha_0 & \alpha_0 \in \text{Prog}_0 \text{ (atomic program)} \\
& \mid & F? & F \text{ formula (test)} \\
& \mid & \alpha; \beta & \text{(sequential composition)} \\
& \mid & \alpha \cup \beta & \text{(non-deterministic choice)} \\
& \mid & \alpha^* & \text{(non-deterministic repetition)}
\end{array}
$$

Semantics

A PDL structure $\mathcal{K} = (S, R(), I)$ is a multimodal Kripke structure with an accessibility relation for each atomic program. That is it consists of:

- a non-empty set S of states
- an interpretation $R() : \mathsf{Prog}_0 \to \mathcal{P}(S \times S)$ of atomic programs that assigns a transition relation $R(\alpha) \subseteq S \times S$ to each atomic program α
- an interpretation $I : \Pi \times S \to \{0, 1\}$

PDL: Semantics

The interpretation of PDL relative to a PDL structure $\mathcal{K} = (S, R(), I)$ is defined by extending $R()$ to Prog and extensing I to $\mathrm{Fma}^{PDL}_{\mathrm{Prop}_0}$ by the following simultaneously inductive definition:

Interpretation of formulae/programs

$$
\begin{array}{lcl}
val_{\mathcal{K}}(p, s) & = & I(p, s) \\
val_{\mathcal{K}}(\neg F, s) & = & \neg_{\mathsf{Bool}} val_{\mathcal{K}}(F, s) \\
val_{\mathcal{K}}(F \wedge G, s) & = & val_{\mathcal{K}}(F, s) \wedge_{\mathsf{Bool}} val_{\mathcal{K}}(G, s) \\
val_{\mathcal{K}}(F \vee G, s) & = & val_{\mathcal{K}}(F, s) \vee_{\mathsf{Bool}} val_{\mathcal{K}}(G, s) \\
val_{\mathcal{K}}([\alpha]F, s) = 1 & \textit{iff} & \text{ for all } t \in S \text{ with } (s, t) \in R(\alpha), val_{\mathcal{K}}(F, t) = 1 \\
val_{\mathcal{K}}(\langle\alpha\rangle\, F, s) = 1 & \textit{iff} & \text{ for some } t \in S \text{ with } (s, t) \in R(\alpha), val_{\mathcal{K}}(F, t) = 1 \\
R([F?]) & = & \{(s, s) \mid val_{\mathcal{K}}(F, s) = 1\} \\
 & & (F\text{? has the same meaning as: if } F \text{ then skip else do not terminate}) \\
R(\alpha \cup \beta) & = & R(\alpha) \cup R(\beta) \\
R(\alpha; \beta) & = & \{(s, t) \mid \text{ there exists } u \in S \text{ } s.t.(s, u) \in R(\alpha) \text{ and } (u, t) \in R(\beta)\} \\
R(\alpha^*) & = & \{(s, t) \mid \text{ there exists } n \geq 0 \text{ and there exist } u_0, \dots, u_n \in S \text{ with} \\
 & & s = u_0, y = u_n, (u_0, u_1), \dots, (u_{n-1}, u_n) \in R(\alpha)\}
\end{array}
$$

Interpretation of formulae/programs

- $(\mathcal{K}, s)$ satisfies F (notation $(\mathcal{K}, s) \models F$) iff $val_{\mathcal{K}}(F, s) = 1$.
- F is valid in $\mathcal{K}$ (notation $\mathcal{K} \models F$) iff $(\mathcal{K}, s) \models F$ for all $s \in S$.
- F is valid (notation $\models F$) iff $\mathcal{K} \models F$ for all PDL-structures $\mathcal{K}$.

Axiom system for PDL

$$\begin{aligned}
&\text{Comp}: && [\alpha;\beta]A \leftrightarrow [\alpha][\beta]A,\\
&\text{Alt}: && [\alpha\cup\beta]A \leftrightarrow [\alpha]A \wedge [\beta]A,\\
&\text{Mix}: && [\alpha^*]A \rightarrow A \wedge [\alpha][\alpha^*]A,\\
&\text{Ind}: && [\alpha^*](A \rightarrow [\alpha]A) \rightarrow (A \rightarrow [\alpha^*]A),\\
&\text{Test}: && [A?]B \leftrightarrow (A \rightarrow B).
\end{aligned}$$

We will show that PDL is determined by PDL structures, and has the finite model property.

Soundness and Completeness of PDL

Proof similar to the proof in the case of the modal system K (with small differences)

Theorem. If the formula F is provable in the inference system for PDL then F is valid in all PDL structures.

Proof: The axioms are valid in every PDL structure. Easy computation (examples on the blackboard).