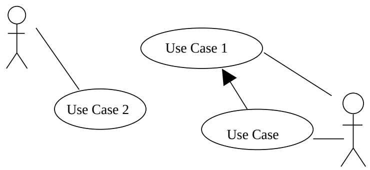
Entwicklung objektorientierter Software mit formalen Methoden

Program Verification – Dynamic Logic for Users

Bernhard Beckert

UNIVERSITÄT KOBLENZ-LANDAU

Verification in different design phases



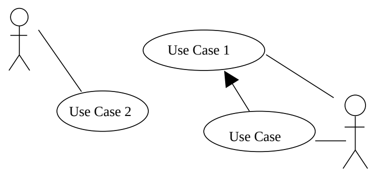
Analyse
Diagrams

+

Requirements
OCL + nat. Language

time

Verification in different design phases



Analyse
Diagrams

+

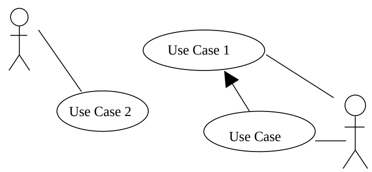
Requirements
OCL + nat. Language

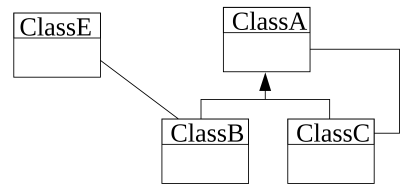
time

—

(semantic gap)

Verification in different design phases





Analyse Diagrams

+

Requirements

OCL + nat. Language

Design Diagrams

+

Specification

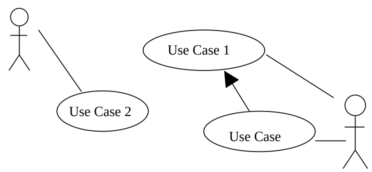
OCL (inv., pre−/post)

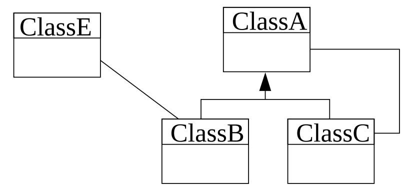
time

—

(semantic gap)

Verification in different design phases





Analyse Diagrams	Design Diagrams
+	+
Requirements OCL + nat. Language	Specification OCL (inv., pre−/post)

Refinement

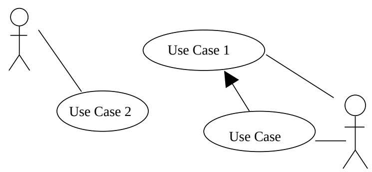
time

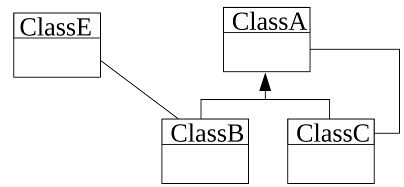
—

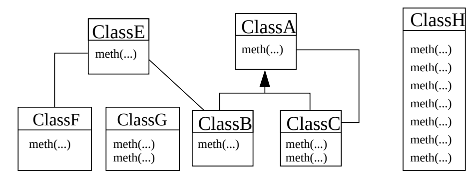
(semantic gap)

Horizontal Verification

Verification in different design phases







Analyse Diagrams	Design Diagrams	Implementation Diagrams
+	+	+
Requirements OCL + nat. Language	Specification OCL (inv., pre−/post)	Source Code Java, C++, Prolog

Refinement

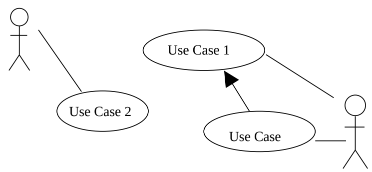
time

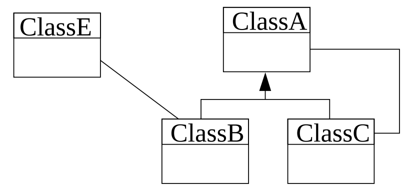
—

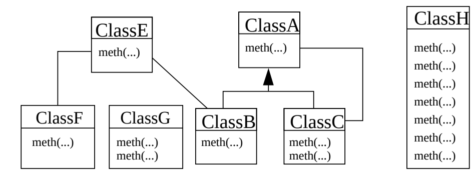
(semantic gap)

Horizontal Verification

Verification in different design phases







Analyse Diagrams	Design Diagrams	Implementation Diagrams
+	+	+
Requirements OCL + nat. Language	Specification OCL (inv., pre−/post)	Source Code Java, C++, Prolog

Refinement

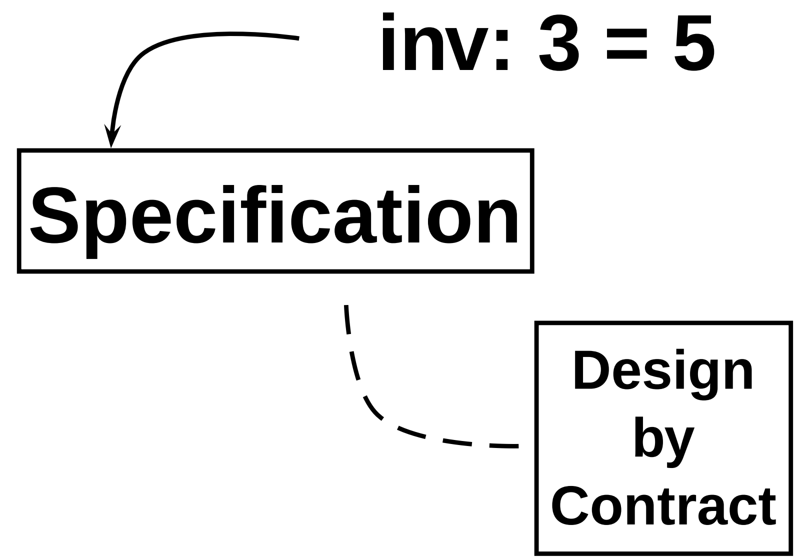
Equivalence

time →

— (semantic gap)	Horizontal Verification	Vertical Verification

What has to be proved?

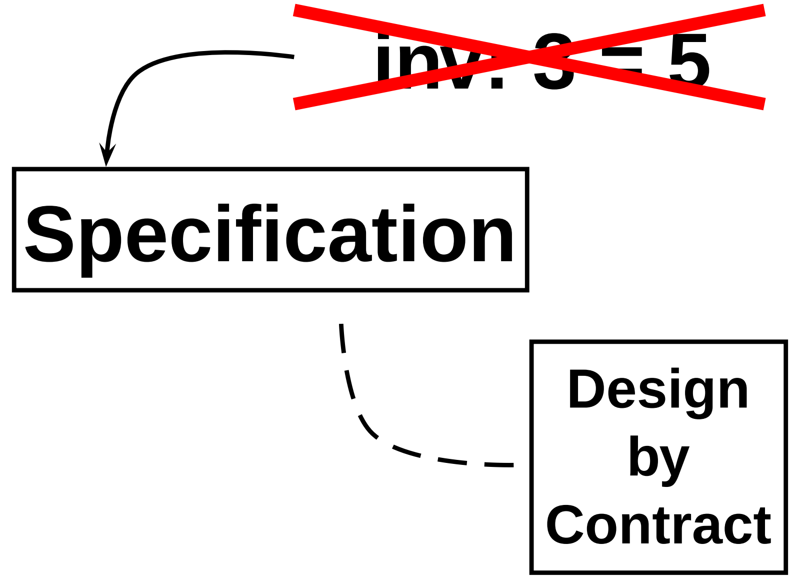
Horizontal Verification



What has to be proved?

Horizontal Verification

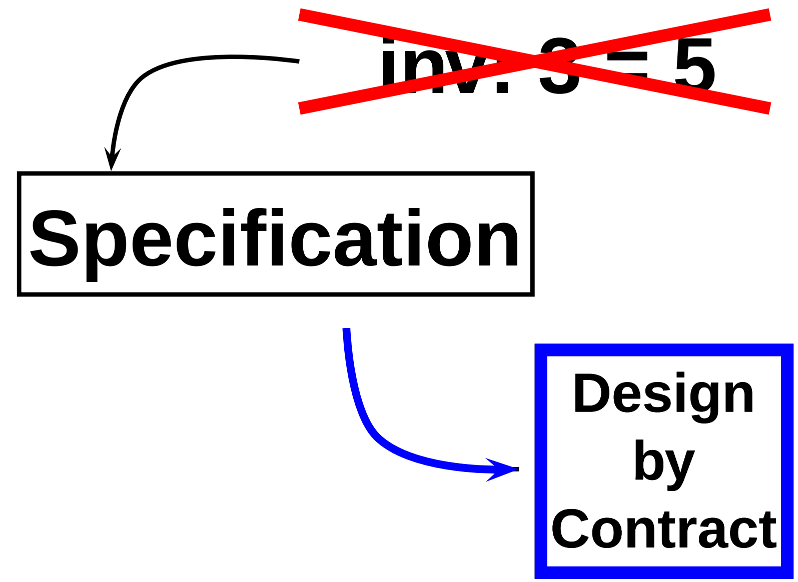
- **Consistency properties**



What has to be proved?

Horizontal Verification

- **Consistency properties**
- **Compliance to design principles**

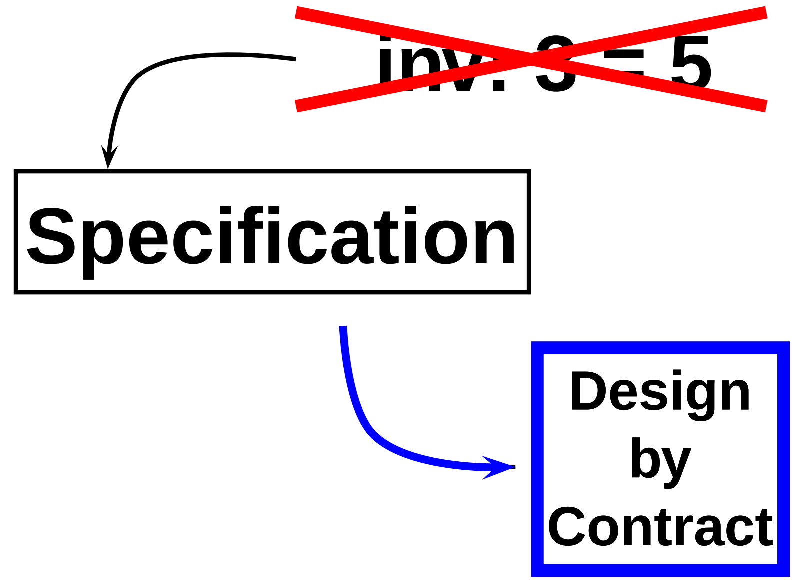


What has to be proved?

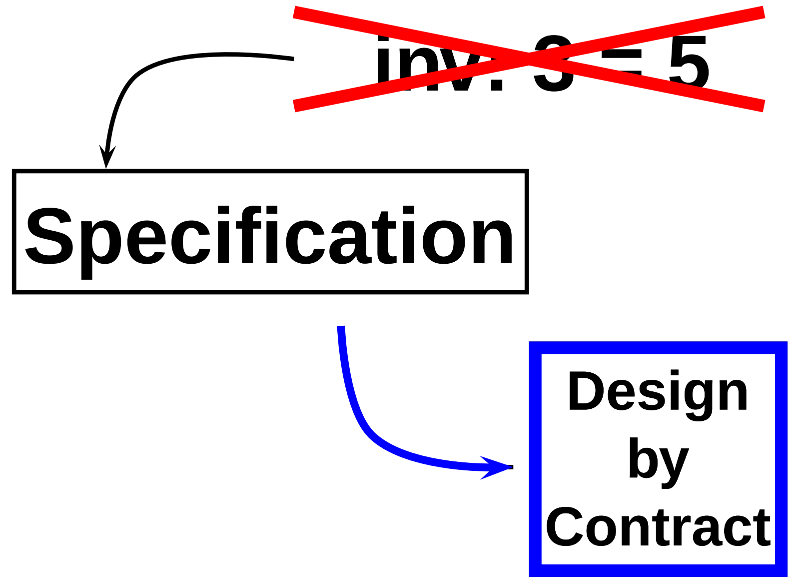
<u>Horizontal Verification</u>

- Consistency properties
- Compliance to design principles

⇒ source code is not involved



What has to be proved?



Horizontal Verification

- Consistency properties
- Compliance to design principles

$\Rightarrow$ source code is not involved

Horizontal Verification can be done in Classical First-Order Logic (FOL)

Syntax of Propositional Logic

Signature $\Sigma = (\mathcal{P}, \mathcal{O})$

- **Propositional Variables** $\mathcal{P} = \{P_i | i \in I\!N\}$

Syntax of Propositional Logic

Signature $\Sigma = (\mathcal{P}, \mathcal{O})$

- **Propositional Variables** $\mathcal{P} = \{P_i | i \in I\!N\}$
- **Logical Operators** $\mathcal{O} = \{\wedge, \vee, \neg\}$ **(handle $\rightarrow$, $\leftrightarrow$ as abbreviations)**

Syntax of Propositional Logic

Signature $\Sigma = (\mathcal{P}, \mathcal{O})$

- **Propositional Variables** $\mathcal{P} = \{P_i | i \in I\!N\}$
- **Logical Operators** $\mathcal{O} = \{\wedge, \vee, \neg\}$ **(handle** $\rightarrow$**,** $\leftrightarrow$ **as abbreviations)**

Formulas For_0^Σ

- **Propositional Variables are formulas**

Syntax of Propositional Logic

Signature $\Sigma = (\mathcal{P}, \mathcal{O})$

- **Propositional Variables** $\mathcal{P} = \{P_i | i \in I\!N\}$
- **Logical Operators** $\mathcal{O} = \{\wedge, \vee, \neg\}$ **(handle** $\rightarrow$**,** $\leftrightarrow$ **as abbreviations)**

Formulas For_0^Σ

- **Propositional Variables are formulas**
- **If** G **and** H **are formulas then**

$$\neg G, (G \wedge H) \text{ and } (G \vee H)$$

are also formulas

Semantics of Propositional Logic

Interpretation (Assignment) I

Assigns a definite truth value to each propositional variable

$$I : \mathcal{P} \rightarrow \{true, false\}$$

Semantics of Propositional Logic

Interpretation (Assignment) I

Assigns a definite truth value to each propositional variable

$$I : \mathcal{P} \rightarrow \{true, false\}$$

Valuation val_I: Continuation of I on For_0^Σ

$$val_I : For_0^\Sigma \rightarrow \{true, false\}$$

Semantics of Propositional Logic

Interpretation (Assignment) I

Assigns a definite truth value to each propositional variable

$$I : \mathcal{P} \rightarrow \{true, false\}$$

Valuation val_I: Continuation of I on For_0^Σ

$$val_I : For_0^\Sigma \;\rightarrow\; \{true, false\}$$

$val_I(P_i) = I(P_i)$

Semantics of Propositional Logic

Interpretation (Assignment) I

Assigns a definite truth value to each propositional variable

$$I : \mathcal{P} \rightarrow \{true, false\}$$

Valuation val_I: Continuation of I on For_0^Σ

$$val_I : For_0^\Sigma \; \rightarrow \; \{true, false\}$$

$$val_I(P_i) = I(P_i) \qquad val_I(P_i \wedge P_j) = \begin{cases} true & \text{if } val_I(P_i) = true \text{ and} \\ & \quad val_I(P_j) = true \\ false & otherwise \end{cases}$$

$\ldots (and\ so\ on)$

»The truth that's me.«, said the tautology.

Let $\Phi \in For_0^\Sigma$, $\Gamma \subset For_0^\Sigma$

- I **is a model for** Φ **iff.** $val_I(\Phi) = true$ **(write:** $I \models \Phi$**)**

»The truth that's me.«, said the tautology.

Let $\Phi \in For_0^\Sigma$, $\Gamma \subset For_0^\Sigma$

- **I is a <u>model</u> for Φ iff. $val_I(\Phi) = true$ (write: $I \models \Phi$)**
- **$\Gamma \models \Phi$ iff. for all interpretations I:**

»The truth that's me.«, said the tautology.

Let $\Phi \in For_0^\Sigma$, $\Gamma \subset For_0^\Sigma$

- I **is a <u>model</u> for** Φ **iff.** $val_I(\Phi) = true$ **(write:** $I \models \Phi$**)**
- $\Gamma \models \Phi$ **iff. for all interpretations** I**:**

$$I \models \Psi \text{ for all } \Psi \in \Gamma \text{ then also } I \models \Phi$$

»The truth that's me.«, said the tautology.

Let $\Phi \in For_0^\Sigma$**,** $\Gamma \subset For_0^\Sigma$

- I **is a** **model** **for** Φ **iff.** $val_I(\Phi) = true$ **(write:** $I \models \Phi$**)**
- $\Gamma \models \Phi$ **iff. for all interpretations** I**:**

$$I \models \Psi \text{ for all } \Psi \in \Gamma \text{ then also } I \models \Phi$$

- **If** Φ **is valid under all interpretations, i.e**

$$\emptyset \models \Phi \ (\textbf{short}: \ \models \Phi)$$

then Φ **is called a** **tautology****.**

Orientation Map

THE SUN SHINES

THE PEOPLE ARE HAPPY

Orientation Map

	THE SUN SHINES	THE PEOPLE ARE HAPPY
Syntax	A	B

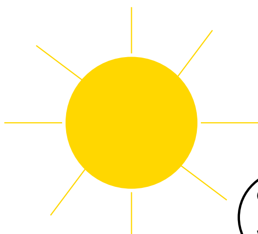
Orientation Map

	IF THE SUN SHINES	THEN	THE PEOPLE ARE HAPPY
Syntax	A	$\longrightarrow$	B

Orientation Map

IF THE SUN SHINES THEN THE PEOPLE ARE HAPPY

Syntax $A \longrightarrow B$

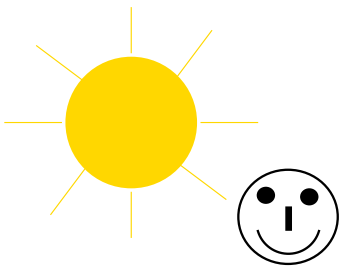
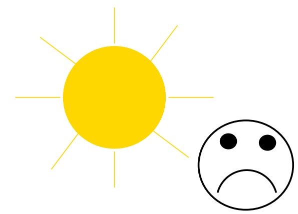


True ☺ ◀ **Semantics** ▶ ☹ **False**

Orientation Map

IF THE SUN SHINES THEN THE PEOPLE ARE HAPPY

Syntax A $\longrightarrow$ B

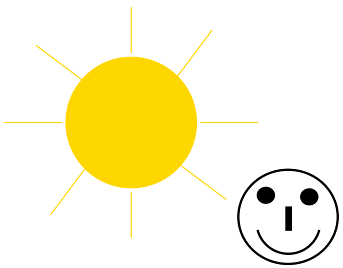
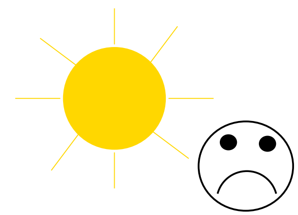
True  ◀ **Semantics** ▶  **False**

Now: Syntactical reasoning

Orientation Map

IF THE SUN SHINES THEN THE PEOPLE ARE HAPPY

Syntax A $\longrightarrow$ B

True  **◂ Semantics ▸**  **False**

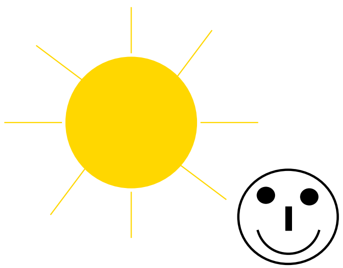
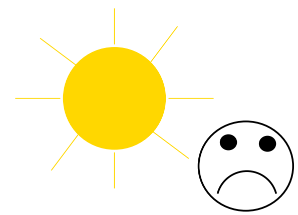
Now: Syntactical reasoning

A THE SUN SHINES

Orientation Map

IF THE SUN SHINES THEN THE PEOPLE ARE HAPPY

Syntax A $\longrightarrow$ B

True  ◀ **Semantics** ▶  **False**

Now: Syntactical reasoning

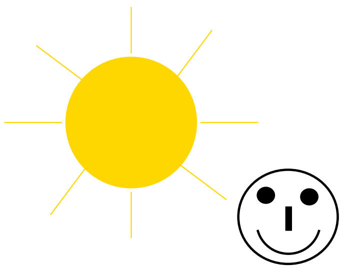
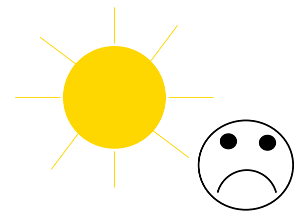
A THE SUN SHINES

$A \rightarrow B$ IF THE SUN SHINES THEN THE PEOPLE ARE HAPPY.

Orientation Map

IF THE SUN SHINES THEN THE PEOPLE ARE HAPPY

Syntax	A	$\longrightarrow$	B

True  ◀ **Semantics** ▶  **False**

Now: Syntactical reasoning

A	THE SUN SHINES
$\underline{A \rightarrow B}$	<u>IF THE SUN SHINES THEN THE PEOPLE ARE HAPPY.</u>
B	THE PEOPLE ARE HAPPY

A Bridge between Semantics and Syntax

Deduction Theorem

Let $\Gamma \subset For_\Sigma$**,** $\Phi, \Psi \in For_\Sigma$

$$\Gamma, \Psi \models \Phi \textbf{ iff. } \Gamma \models \Psi \to \Phi$$

Establishes a relationship between the semantical consequence '$\models$'

and the syntactical implication '$\to$'

Reasoning as Syntactical Transformations

Task: Compute $\Gamma \models \Phi$ by performing syntactical transformations

Reasoning as Syntactical Transformations

Task: Compute $\Gamma \models \Phi$ by performing syntactical transformations

Solution: Calculus $\vdash$ and a set of rules $\mathcal{R}$

Reasoning as Syntactical Transformations

Task: Compute $\Gamma \models \Phi$ by performing syntactical transformations

Solution: Calculus $\vdash$ and a set of rules $\mathcal{R}$

Sequent Calculus '$\Longrightarrow$':

$$\underbrace{\psi_1, \ldots, \psi_n}_{Premises} \Longrightarrow \underbrace{\phi_1, \ldots, \phi_n}_{Consequences}$$

Reasoning as Syntactical Transformations

Task: Compute $\Gamma \models \Phi$ by performing syntactical transformations

Solution: Calculus $\vdash$ and a set of rules $\mathcal{R}$

Sequent Calculus '$\Longrightarrow$':

$$\underbrace{\psi_1, \ldots, \psi_n}_{Premises} \Longrightarrow \underbrace{\phi_1, \ldots, \phi_n}_{Consequences}$$

has the same semantic as

$$\psi_1 \wedge \ldots \wedge \psi_n \rightarrow \phi_1 \vee \ldots \vee \phi_n$$

Rules of the Sequent Calculus

	left side	right side
not	$\dfrac{\Gamma \implies A, \Delta}{\Gamma, \neg A \implies \Delta}$	$\dfrac{\Gamma, A \implies \Delta}{\Gamma \implies \neg A, \Delta}$

Rules of the Sequent Calculus

	left side	right side
not	$\dfrac{\Gamma \implies A, \Delta}{\Gamma, \neg A \implies \Delta}$	$\dfrac{\Gamma, A \implies \Delta}{\Gamma \implies \neg A, \Delta}$
and	$\dfrac{\Gamma, A, B \implies \Delta}{\Gamma, A \wedge B \implies \Delta}$	$\dfrac{\Gamma \implies A, \Delta \qquad \Gamma \implies B, \Delta}{\Gamma \implies A \wedge B, \Delta}$

Rules of the Sequent Calculus

	left side	right side
not	$\dfrac{\Gamma \Longrightarrow A, \Delta}{\Gamma, \neg A \Longrightarrow \Delta}$	$\dfrac{\Gamma, A \Longrightarrow \Delta}{\Gamma \Longrightarrow \neg A, \Delta}$
and	$\dfrac{\Gamma, A, B \Longrightarrow \Delta}{\Gamma, A \wedge B \Longrightarrow \Delta}$	$\dfrac{\Gamma \Longrightarrow A, \Delta \qquad \Gamma \Longrightarrow B, \Delta}{\Gamma \Longrightarrow A \wedge B, \Delta}$
or	$\dfrac{\Gamma, A \Longrightarrow \Delta \qquad \Gamma, B \Longrightarrow \Delta}{\Gamma, A \vee B \Longrightarrow \Delta}$	$\dfrac{\Gamma \Longrightarrow A, B, \Delta}{\Gamma \Longrightarrow A \vee B, \Delta}$

Rules of the Sequent Calculus

	left side	right side
not	$\dfrac{\Gamma \implies A, \Delta}{\Gamma, \neg A \implies \Delta}$	$\dfrac{\Gamma, A \implies \Delta}{\Gamma \implies \neg A, \Delta}$
and	$\dfrac{\Gamma, A, B \implies \Delta}{\Gamma, A \wedge B \implies \Delta}$	$\dfrac{\Gamma \implies A, \Delta \qquad \Gamma \implies B, \Delta}{\Gamma \implies A \wedge B, \Delta}$
or	$\dfrac{\Gamma, A \implies \Delta \qquad \Gamma, B \implies \Delta}{\Gamma, A \vee B \implies \Delta}$	$\dfrac{\Gamma \implies A, B, \Delta}{\Gamma \implies A \vee B, \Delta}$
imp	$\dfrac{\Gamma \implies A, \Delta \quad \Gamma, B \implies \Delta}{\Gamma, A \to B \implies \Delta}$	$\dfrac{\Gamma, A \implies B, \Delta}{\Gamma \implies A \to B, \Delta}$

Rules of the Sequent Calculus

	left side	right side
not	$\dfrac{\Gamma \Longrightarrow A, \Delta}{\Gamma, \neg A \Longrightarrow \Delta}$	$\dfrac{\Gamma, A \Longrightarrow \Delta}{\Gamma \Longrightarrow \neg A, \Delta}$
and	$\dfrac{\Gamma, A, B \Longrightarrow \Delta}{\Gamma, A \wedge B \Longrightarrow \Delta}$	$\dfrac{\Gamma \Longrightarrow A, \Delta \qquad \Gamma \Longrightarrow B, \Delta}{\Gamma \Longrightarrow A \wedge B, \Delta}$
or	$\dfrac{\Gamma, A \Longrightarrow \Delta \qquad \Gamma, B \Longrightarrow \Delta}{\Gamma, A \vee B \Longrightarrow \Delta}$	$\dfrac{\Gamma \Longrightarrow A, B, \Delta}{\Gamma \Longrightarrow A \vee B, \Delta}$
imp	$\dfrac{\Gamma \Longrightarrow A, \Delta \quad \Gamma, B \Longrightarrow \Delta}{\Gamma, A \rightarrow B \Longrightarrow \Delta}$	$\dfrac{\Gamma, A \Longrightarrow B, \Delta}{\Gamma \Longrightarrow A \rightarrow B, \Delta}$

CLOSE(AXIOM) $\dfrac{*}{\Gamma, A \Longrightarrow A, \Delta}$

Proof of Modus Ponens

$$\Gamma \implies (A \land (A \to B)) \to B, \Delta$$

Proof of Modus Ponens

$$\Gamma, (A \wedge (A \to B)) \implies B, \Delta$$

$$\Gamma \implies (A \wedge (A \to B)) \to B, \Delta$$

Proof of Modus Ponens

$$\frac{\dfrac{\dfrac{\overline{} \qquad \overline{}}{\Gamma, A, (A \to B) \Longrightarrow B, \Delta}}{\Gamma, (A \land (A \to B)) \Longrightarrow B, \Delta}}{\Gamma \Longrightarrow (A \land (A \to B)) \to B, \Delta}$$

Proof of Modus Ponens

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma, A \implies B, A, \Delta} \qquad \dfrac{}{\Gamma, A, B \implies B, \Delta}}{\Gamma, A, (A \to B) \implies B, \Delta}}{\Gamma, (A \wedge (A \to B)) \implies B, \Delta}}{\Gamma \implies (A \wedge (A \to B)) \to B, \Delta}$$

Proof of Modus Ponens

$$\dfrac{\dfrac{\dfrac{\dfrac{*}{\Gamma, A \implies B, A, \Delta} \qquad \dfrac{*}{\Gamma, A, B \implies B, \Delta}}{\Gamma, A, (A \to B) \implies B, \Delta}}{\Gamma, (A \land (A \to B)) \implies B, \Delta}}{\Gamma \implies (A \land (A \to B)) \to B, \Delta}$$

Proof of Modus Ponens

$$\dfrac{\dfrac{\dfrac{\dfrac{*}{\Gamma, A \implies B, A, \Delta} \qquad \dfrac{*}{\Gamma, A, B \implies B, \Delta}}{\Gamma, A, (A \to B) \implies B, \Delta}}{\Gamma, (A \land (A \to B)) \implies B, \Delta}}{\Gamma \implies (A \land (A \to B)) \to B, \Delta}$$

A proof is closed, if all its goals are closed.

Propositional logic is insufficient

A ALL PERSONS ARE HAPPY

Propositional logic is insufficient

A	**ALL PERSONS ARE HAPPY**
B	**PAT IS A PERSON**

Propositional logic is insufficient

A	ALL PERSONS ARE HAPPY
B	PAT IS A PERSON
$?$	PAT IS HAPPY

Propositional Logic lacks a possibility to talk about individuals.

Propositional logic is insufficient

A	ALL PERSONS ARE HAPPY
B	PAT IS A PERSON
$?$	PAT IS HAPPY

Propositional Logic lacks a possibility to talk about individuals.

$\Rightarrow$ First-Order Logic (FOL)

Syntax of First-Order Logic

Signature $\Sigma = (\mathcal{P}, \mathcal{F}, \mathcal{V}, \alpha, \mathcal{O} \cup \mathcal{Q} \cup \{\doteq\})$

Syntax of First-Order Logic

Signature $\Sigma = (\mathcal{P}, \mathcal{F}, \mathcal{V}, \alpha, \mathcal{O} \cup \mathcal{Q} \cup \{\doteq\})$

- **Predicate Symbols** $\mathcal{P} = \{P_i | i \in I\!N\}$,
  Function Symbols $\mathcal{F} = \{f_i | i \in I\!N\}$,
  $\Big\}\ \alpha : \mathcal{P} \cup \mathcal{F} \to I\!N$ **(arity)**

Variables $\mathcal{V} = \{x_i | i \in I\!N\}$

Syntax of First-Order Logic

Signature $\Sigma = (\mathcal{P}, \mathcal{F}, \mathcal{V}, \alpha, \mathcal{O} \cup \mathcal{Q} \cup \{\doteq\})$

- **Predicate Symbols** $\mathcal{P} = \{P_i | i \in I\!N\}$,
  Function Symbols $\mathcal{F} = \{f_i | i \in I\!N\}$,
  $\Big\}$ $\alpha : \mathcal{P} \cup \mathcal{F} \to I\!N$ **(arity)**

  Variables $\mathcal{V} = \{x_i | i \in I\!N\}$

- **Operators** $\mathcal{O} = \{\wedge, \vee, \neg\}$, **Quantifiers** $\mathcal{Q} = \{\forall, \exists\}$ **and the syntactical equality** $\doteq$

Syntax of First-Order Logic

Signature $\Sigma = (\mathcal{P}, \mathcal{F}, \mathcal{V}, \alpha, \mathcal{O} \cup \mathcal{Q} \cup \{\doteq\})$

- $$\left.\begin{array}{ll}\textbf{Predicate Symbols} & \mathcal{P} = \{P_i | i \in I\!N\}, \\ \textbf{Function Symbols} & \mathcal{F} = \{f_i | i \in I\!N\},\end{array}\right\} \alpha : \mathcal{P} \cup \mathcal{F} \to I\!N \textbf{ (arity)}$$

  Variables $\mathcal{V} = \{x_i | i \in I\!N\}$

- **Operators** $\mathcal{O} = \{\wedge, \vee, \neg\}$**, Quantifiers** $\mathcal{Q} = \{\forall, \exists\}$ **and the syntactical equality** $\doteq$

Terms $Term_\Sigma$ **and Formulas** For_Σ **are defined inductively as usual.**

Syntax of First-Order Logic

Signature $\Sigma = (\mathcal{P}, \mathcal{F}, \mathcal{V}, \alpha, \mathcal{O} \cup \mathcal{Q} \cup \{\doteq\})$

- **Predicate Symbols** $\mathcal{P} = \{P_i | i \in I\!N\}$, **Function Symbols** $\mathcal{F} = \{f_i | i \in I\!N\}$, $\Big\}$ $\alpha : \mathcal{P} \cup \mathcal{F} \to I\!N$ **(arity)**

  Variables $\mathcal{V} = \{x_i | i \in I\!N\}$

- **Operators** $\mathcal{O} = \{\wedge, \vee, \neg\}$, **Quantifiers** $\mathcal{Q} = \{\forall, \exists\}$ **and the syntactical equality** $\doteq$

Terms $Term_\Sigma$ **and Formulas** For_Σ **are defined inductively as usual.**

Additional: Let t_1, t_2 **be terms then** $t_1 \doteq t_2$ **is a formula.**

Semantics of First-Order Logic

Interpretation $\mathcal{D}$=(U, I):

U **is the non-empty universe**

Semantics of First-Order Logic

Interpretation $\mathcal{D}$=(U, I):

- U **is the non-empty universe**

  $P^I \subseteq \{(x_1, \ldots, x_n) | x_i \in U, n = \alpha(P)\}$

  $f^I : U^{\alpha(f)} \to U$

Semantics of First-Order Logic

Interpretation $\mathcal{D}$=(U, I):

U **is the non-empty universe**

$$P^I \subseteq \{(x_1, \ldots, x_n) | x_i \in U, n = \alpha(P)\}$$

$$f^I : U^{\alpha(f)} \rightarrow U$$

Variable Assignment $\beta : \mathcal{V} \rightarrow U$

$$val_{\mathcal{D},\beta}(P(x_1, \ldots, x_n)) = \begin{cases} true & (\beta(x_1), ..., \beta(x_n)) \in P^I \\ false & otherwise \end{cases}$$

Semantics of First-Order Logic

Interpretation $\mathcal{D}$=(U, I):

- U **is the non-empty universe**

  $P^I \subseteq \{(x_1, \ldots, x_n) | x_i \in U, n = \alpha(P)\}$

  $f^I : U^{\alpha(f)} \rightarrow U$

Variable Assignment $\beta : \mathcal{V} \rightarrow U$

$$val_{\mathcal{D},\beta}(P(x_1, \ldots, x_n)) = \begin{cases} true & (\beta(x_1), ..., \beta(x_n)) \in P^I \\ false & otherwise \end{cases}$$

$$val_{\mathcal{D},\beta}(\forall x.\Phi(x)) = \begin{cases} true & \textbf{for all } d \in U : val_{\mathcal{D},\beta_x^d}(\Phi) = true \\ false & otherwise \end{cases}$$

Definitions

Satisfiability, Model and Universal validity

$\mathcal{D}, \beta \models \Phi$ **iff.** $val_{\mathcal{D},\beta}(\Phi) = true$ (Φ **is satisfiable**)

Definitions

Satisfiability, Model and Universal validity

$\mathcal{D}, \beta \models \Phi$ **iff.** $val_{\mathcal{D},\beta}(\Phi) = true$ (Φ **is satisfiable**)

$\mathcal{D} \models \Phi$ **iff.** **for all** β : $\mathcal{D}, \beta \models \Phi$ (Φ **is valid**)

Definitions

Satisfiability, Model and Universal validity

$\mathcal{D}, \beta \models \Phi$	**iff.**	$val_{\mathcal{D},\beta}(\Phi) = true$	(Φ **is satisfiable**)
$\mathcal{D} \quad \models \Phi$	**iff.**	**for all** β : $\mathcal{D}, \beta \models \Phi$	(Φ **is valid**)
$\models \Phi$	**iff.**	**for all** $\mathcal{D}$: $\mathcal{D} \models \Phi$	(Φ **is universally valid**)

Definitions

Satisfiability, Model and Universal validity

$\mathcal{D}, \beta \models \Phi$ **iff.** $val_{\mathcal{D},\beta}(\Phi) = true$ (Φ **is satisfiable**)

$\mathcal{D} \models \Phi$ **iff.** **for all** $\beta : \mathcal{D}, \beta \models \Phi$ (Φ **is valid**)

$\models \Phi$ **iff.** **for all** $\mathcal{D} : \mathcal{D} \models \Phi$ (Φ **is universally valid**)

REMARK: Sorted First-Order Logic

Variables and functions is given a sort $\in Sorts$

$\forall x : S.\Phi(x)$ **i.e.** $\forall x.(S(x) \rightarrow \Phi(x))$

$\exists x : S.\Phi(x)$ **i.e.** $\exists x.(S(x) \wedge \Phi(x))$

Do we have a deduction theorem at hand?

$$\Gamma, \Psi \models \Phi \text{ **iff.** } \Gamma \models \Psi \to \Phi$$

?

Do we have a deduction theorem at hand?

$\Gamma, \Psi \models \Phi$ **iff.** $\Gamma \models \Psi \to \Phi$

?

Yes, but only if Ψ is closed.

Do we have a deduction theorem at hand?

$$\Gamma, \Psi \models \Phi \textbf{ iff. } \Gamma \models \Psi \to \Phi$$

?

Yes, but only if Ψ is closed.

From now on only closed formulas are considered.

Sequent Calculus for FOL

	left side	right side

- $t \in Term_\Sigma$ **an arbitrary ground term (no variables)**
- $c\ new$ **constant**

Sequent Calculus for FOL

	left side	right side
all	$\dfrac{\Gamma, \forall x.\Phi(x), \{x/t\}\Phi(x) \implies \Delta}{\Gamma, \forall x.\Phi(x) \implies \Delta}$	$\dfrac{\Gamma \implies \{x/c\}\Phi(x), \Delta}{\Gamma \implies \forall x.\Phi(x), \Delta}$

- $t \in Term_\Sigma$ **an arbitrary ground term (no variables)**
- $c\ new$ **constant**

Sequent Calculus for FOL

	left side	right side
all	$$\frac{\Gamma, \forall x.\Phi(x), \{x/t\}\Phi(x) \implies \Delta}{\Gamma, \forall x.\Phi(x) \implies \Delta}$$	$$\frac{\Gamma \implies \{x/c\}\Phi(x), \Delta}{\Gamma \implies \forall x.\Phi(x), \Delta}$$
ex.	$$\frac{\Gamma \implies \{x/t\}\Phi(x), \exists x.\Phi(x), \Delta}{\Gamma \implies \exists x.\Phi(x), \Delta}$$	$$\frac{\Gamma, \{x/c\}\Phi(x) \implies \Delta}{\Gamma, \exists x.\Phi(x) \implies \Delta}$$

- $t \in Term_\Sigma$ **an arbitrary ground term (no variables)**
- $c\ new$ **constant**

Sequent Calculus for FOL

	left side	right side
all	$\dfrac{\Gamma, \forall x.\Phi(x), \{x/t\}\Phi(x) \implies \Delta}{\Gamma, \forall x.\Phi(x) \implies \Delta}$	$\dfrac{\Gamma \implies \{x/c\}\Phi(x), \Delta}{\Gamma \implies \forall x.\Phi(x), \Delta}$
ex.	$\dfrac{\Gamma \implies \{x/t\}\Phi(x), \exists x.\Phi(x), \Delta}{\Gamma \implies \exists x.\Phi(x), \Delta}$	$\dfrac{\Gamma, \{x/c\}\Phi(x) \implies \Delta}{\Gamma, \exists x.\Phi(x) \implies \Delta}$
insert eq.	$\dfrac{\Gamma, x \doteq y \implies \{x/y\}\Phi(x), \Delta}{\Gamma, x \doteq y \implies \Phi(x), \Delta}$	—

- $t \in Term_{\Sigma}$ **an arbitrary ground term (no variables)**
- $c\ new$ **constant**

Explaining the Rules (I)

The following description shall explain the first-order calculus rules on an intuitive (informal) level. For the remainding section all mentioned terms are ground terms, this means they contain no variables.

`all left` If a $\forall x.\Phi(x)$ occurs in the premise, one can add an instantiation with an arbitrary term t to the premises. This is sound as $\{x/t\}\Phi(x)$ holds for all elements of the universe, in particular for the element t is evaluated to. In contrast to the former rules one keeps the quantified formula in the antecedent as one may require more than one instantiation.

`ex. left` $\exists x.\Phi(x)$ can be replaced by $\{x/c\}\Phi$ where c is a new constant. c is thought to be evaluated to the element for which $\Phi(x)$ holds. An already existing term t must not be used as its value is already fixed but in general not to the element satisfying $\Phi(x)$.

Explaining the Rules (II)

`all right` A common way to show that $\forall x.\Phi(x)$ holds, is to take an element of an arbitrary value. In other words, if $\{x/c\}\Phi(x)$ can be shown for a new constant c then the result can be generalised, as no assumptions about the value of c have been made.
In contrast, the generalisation is not possible if an already existing term t is used instead. The value of t has been already fixed to a certain value, which may randomly satisfy $\Phi(x)$, but this may not necessarily be the case for all other elements of the universe (similar to: $2, 3, 5, 7$ are primes, so all odd numbers are primes).

`ex. right` If $\exists x.\Phi(x)$ has to be proven, one can try to prove it for an arbitrary term t. If one uses the wrong term t, this means a term for which $\Phi(x)$ is *false* it is not worse, one only gets *false* on the right side, which is the neutral element of $\vee$ and so it can just be removed from the sequent. The existential quantified formula is not removed from the sequent, so that one can try to prove the formula for another term t' (sometimes one even has to instantiate the existential quantifiers and all instances are required).

Example

DEMO

Towards Program Verification

Vertical Verification

- **Prove that the implementation fulfills the specification (equivalence for complete specifications)**

Towards Program Verification

Vertical Verification

- **Prove that the implementation fulfills the specification (equivalence for complete specifications)**
- **Reasoning about programs**

Towards Program Verification

Vertical Verification

- **Prove that the implementation fulfills the specification (equivalence for complete specifications)**
- **Reasoning about programs**
- **Formalise program properties as formulas of Dynamic Logic**

Towards Program Verification

Vertical Verification

- **Prove that the implementation fulfills the specification (equivalence for complete specifications)**
- **Reasoning about programs**
- **Formalise program properties as formulas of Dynamic Logic**

In contrast to testing,
verification can show the absence of errors

Do we really need another kind of logics?

»There is a tradition in logic, carried over into computer science, to think of pure first order logic as a universal language.
In fact first order language is about as useful in verification as a Turing machine is in software engineering:

CUTE TO WATCH BUT NOT VERY USEFUL.«

V. Pratt

State Dependance of Truth Values

What is the truth value of

? 'The value of program variable x is 3.' ?

State Dependance of Truth Values

What is the truth value of

? 'The value of program variable x is 3.' ?

May vary during the execution time of a program.

State Dependance of Truth Values

What is the truth value of

? 'The value of program variable x is 3.' ?

May vary during the execution time of a program.

For example, after the execution of

- `x=3;` **the value is** $true$

State Dependance of Truth Values

What is the truth value of

? 'The value of program variable x is 3.' ?

May vary during the execution time of a program.

For example, after the execution of

- `x=3;` **the value is** $true$
- `x=4;` **the value is** $false$

State Dependance of Truth Values

What is the truth value of

? 'The value of program variable x is 3.' ?

May vary during the execution time of a program.

For example, after the execution of

- `x=3;` **the value is** $true$
- `x=4;` **the value is** $false$

$\Rightarrow$ **Reasoning about programs must consider the current program state.**

Dynamic Logics for a simple 'while' language

Signature

$\Sigma = (\mathcal{P}, \mathcal{F}, \Pi_0, \mathcal{O} \cup \{\langle\cdot\rangle, [.]\})$, $Sorts = \{\texttt{int}, \texttt{boolean}\}$

Dynamic Logics for a simple 'while' language

Signature

$\Sigma = (\mathcal{P}, \mathcal{F}, \Pi_0, \mathcal{O} \cup \{\langle\cdot\rangle, [.]\})$, $Sorts = \{\mathtt{int}, \mathtt{boolean}\}$

Π_0 **is a set of atomic programs (e.g.** α, β**)**

Dynamic Logics for a simple 'while' language

Signature

$\Sigma = (\mathcal{P}, \mathcal{F}, \Pi_0, \mathcal{O} \cup \{\langle\cdot\rangle, [.]\})$, $Sorts = \{\texttt{int}, \texttt{boolean}\}$

Π_0 **is a set of atomic programs (e.g.** α, β**)**

Definition of Programs Π

Dynamic Logics for a simple 'while' language

Signature

$\Sigma = (\mathcal{P}, \mathcal{F}, \Pi_0, \mathcal{O} \cup \{\langle\cdot\rangle, [.]\})$, $Sorts = \{\texttt{int}, \texttt{boolean}\}$

Π_0 **is a set of atomic programs (e.g.** α, β**)**

Definition of Programs Π

If $\alpha, \beta \in \Pi_0$ **and** b **a term of sort** `bool` **then**

- $\alpha; \beta$

Dynamic Logics for a simple 'while' language

Signature

$\Sigma = (\mathcal{P}, \mathcal{F}, \Pi_0, \mathcal{O} \cup \{\langle\cdot\rangle, [.]\})$, $Sorts = \{\texttt{int}, \texttt{boolean}\}$

Π_0 **is a set of atomic programs (e.g.** α, β**)**

Definition of Programs Π

If $\alpha, \beta \in \Pi_0$ **and** b **a term of sort** `bool` **then**

- $\alpha; \beta$
- `if` (b) `then` $\{\alpha\}$ `else` $\{\beta\}$

Dynamic Logics for a simple 'while' language

Signature

$\Sigma = (\mathcal{P}, \mathcal{F}, \Pi_0, \mathcal{O} \cup \{\langle\cdot\rangle, [.]\})$, $Sorts = \{\texttt{int}, \texttt{boolean}\}$

Π_0 **is a set of atomic programs (e.g.** α, β**)**

Definition of Programs Π

If $\alpha, \beta \in \Pi_0$ **and** b **a term of sort** `bool` **then**

- $\alpha; \beta$
- $\texttt{if } (b) \texttt{ then } \{\alpha\} \texttt{ else } \{\beta\}$
- $\texttt{while } (b)\{\alpha\}$

are programs in Π**.**

Terms and Formulas of Dynamic Logics

Definition of Terms

Defined as in first-order logics. But we distinct between

- **rigid terms, which are meant to be state independant**

Terms and Formulas of Dynamic Logics

Definition of Terms

Defined as in first-order logics. But we distinct between

- **rigid terms, which are meant to be state independant**
- **non-rigid (or flexible) terms, whose value (interpretation) will depend on the current program state**

Terms and Formulas of Dynamic Logics

Definition of Terms

Defined as in first-order logics. But we distinct between

- **rigid terms, which are meant to be state independant**
- **non-rigid (or flexible) terms, whose value (interpretation) will depend on the current program state**

Definition of Formulas

All formulas of FOL are also dynamic logic formulas (DL formulas).

Terms and Formulas of Dynamic Logics

Definition of Terms

Defined as in first-order logics. But we distinct between

- rigid terms, which are meant to be state independant
- non-rigid (or flexible) terms, whose value (interpretation) will depend on the current program state

Definition of Formulas

All formulas of FOL are also dynamic logic formulas (DL formulas).

If α is a program and Φ a formula then

Terms and Formulas of Dynamic Logics

Definition of Terms

Defined as in first-order logics. But we distinct between

- rigid terms, which are meant to be state independant
- non-rigid (or flexible) terms, whose value (interpretation) will depend on the current program state

Definition of Formulas

All formulas of FOL are also dynamic logic formulas (DL formulas).

If α is a program and Φ a formula then

$\langle\alpha\rangle\Phi$ is a DL-Formula

Terms and Formulas of Dynamic Logics

Definition of Terms

Defined as in first-order logics. But we distinct between

- **rigid terms, which are meant to be state independant**
- **non-rigid (or flexible) terms, whose value (interpretation) will depend on the current program state**

Definition of Formulas

All formulas of FOL are also dynamic logic formulas (DL formulas).

If α is a program and Φ a formula then

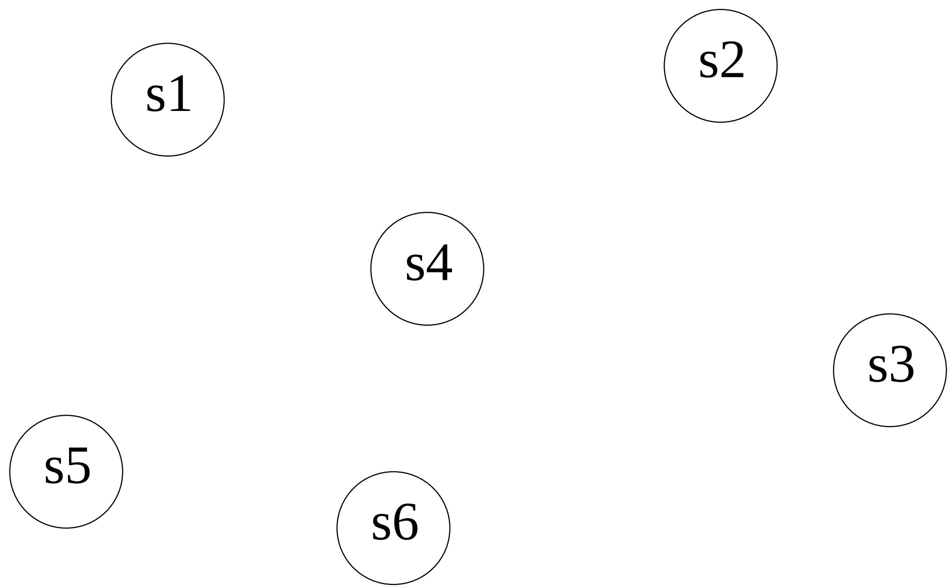
$\langle\alpha\rangle\Phi$ **is a DL-Formula**

$[\alpha]\Phi$ **is a DL-Formula**

Semantics of Dynamic Logic - Kripke Structure

Kripke-Structure $\mathcal{K} = (States, \rho)$

where $s \in State, s = (\mathcal{U}, I)$ **and** $\rho : \Pi_0 \rightarrow States \times States$

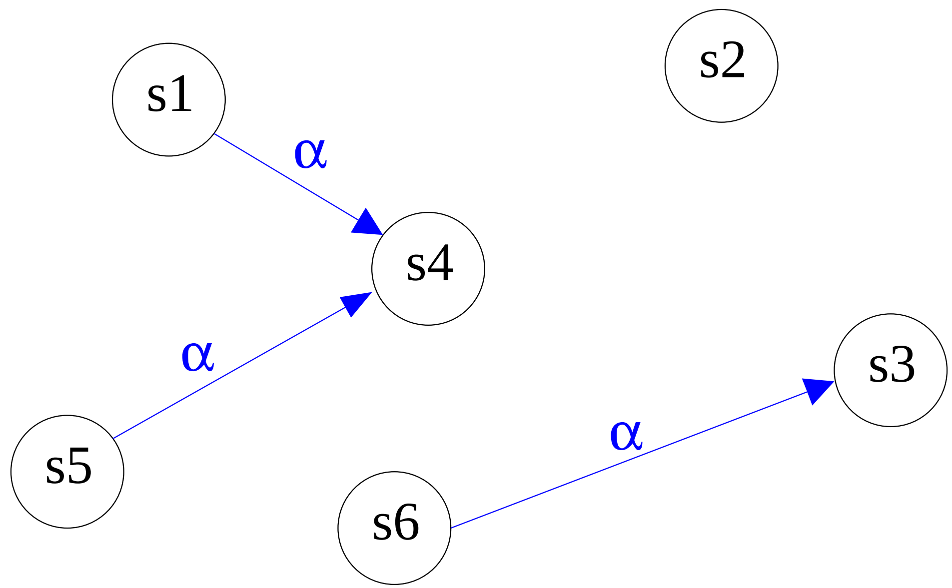


Semantics of Dynamic Logic - Kripke Structure

Kripke-Structure $\mathcal{K} = (States, \rho)$

where $s \in State, s = (\mathcal{U}, I)$ **and** $\rho : \Pi_0 \to States \times States$

$\rho(\alpha)$

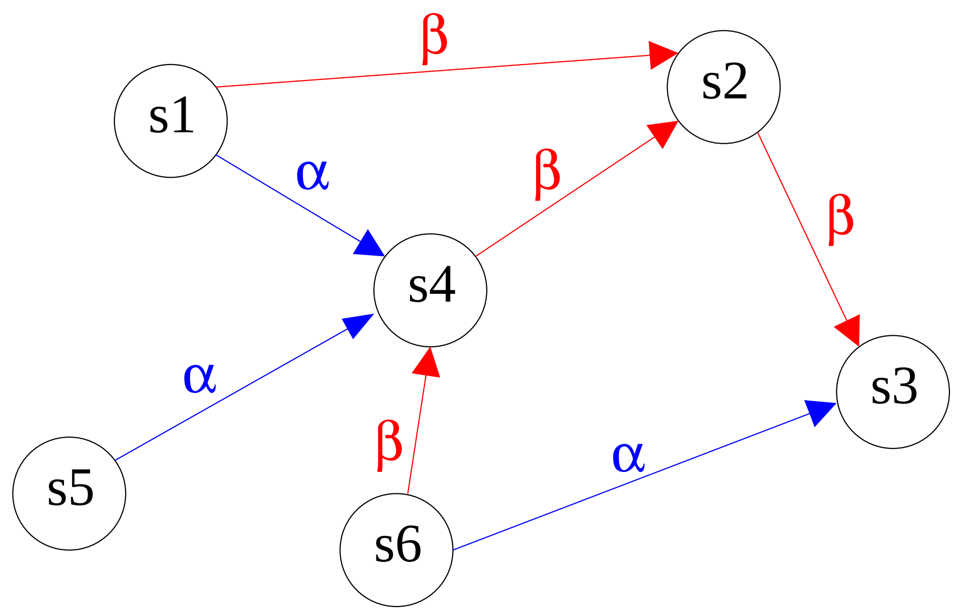


Semantics of Dynamic Logic - Kripke Structure

Kripke-Structure $\mathcal{K} = (States, \rho)$

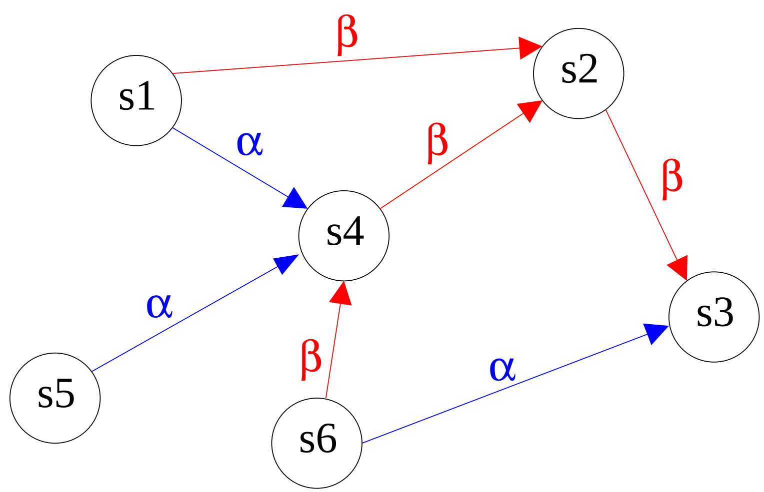
where $s \in State, s = (\mathcal{U}, I)$ **and** $\rho : \Pi_0 \to States \times States$

$\rho(\alpha)$, $\rho(\beta)$



Diamond and Box Revealed

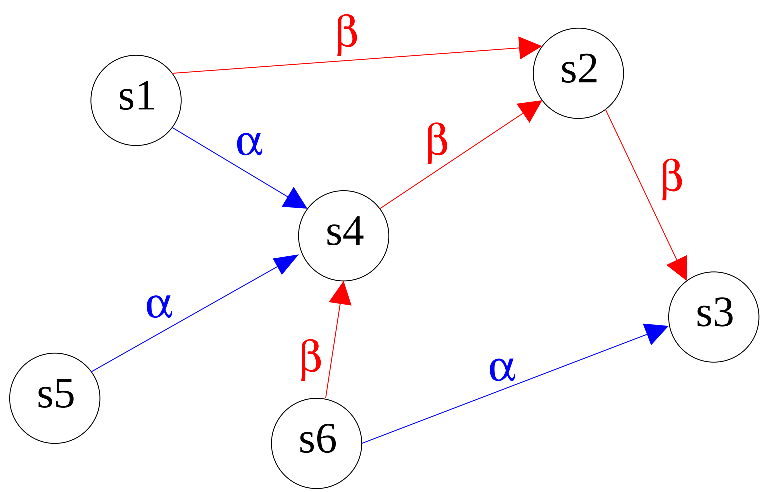
$\langle\alpha\rangle\Phi$ **There exists an α-reachable state, such that Φ holds.**



Diamond and Box Revealed

$\langle\alpha\rangle\Phi$ **There exists an α-reachable state, such that Φ holds.**

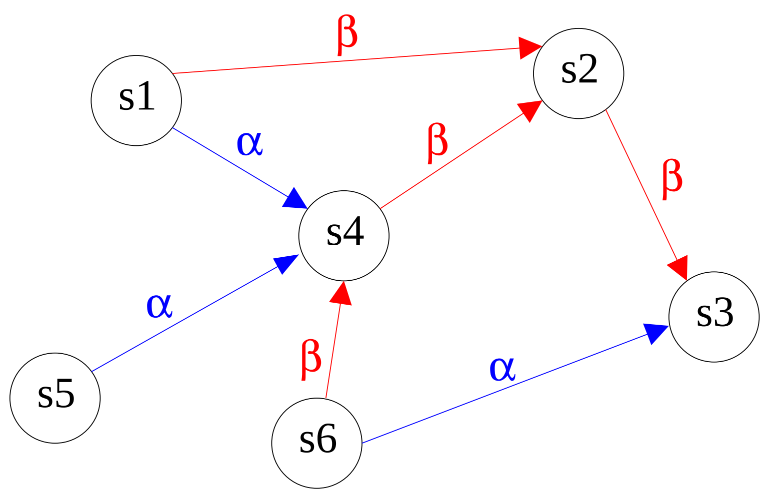
$[\alpha]\Phi$ **Φ holds in all α-reachable states.**



Diamond and Box Revealed

$\langle\alpha\rangle\Phi$ **There exists an α-reachable state, such that Φ holds.**

$[\alpha]\Phi$ **Φ holds in all α-reachable states.**

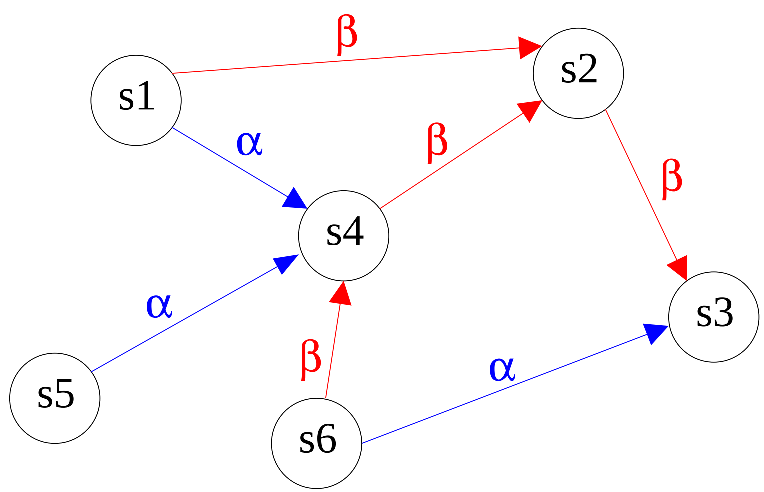


What does this mean in terms of program execution?

Diamond and Box Revealed

$\langle\alpha\rangle\Phi$ **There exists an α-reachable state, such that Φ holds.**

$[\alpha]\Phi$ **Φ holds in all α-reachable states.**



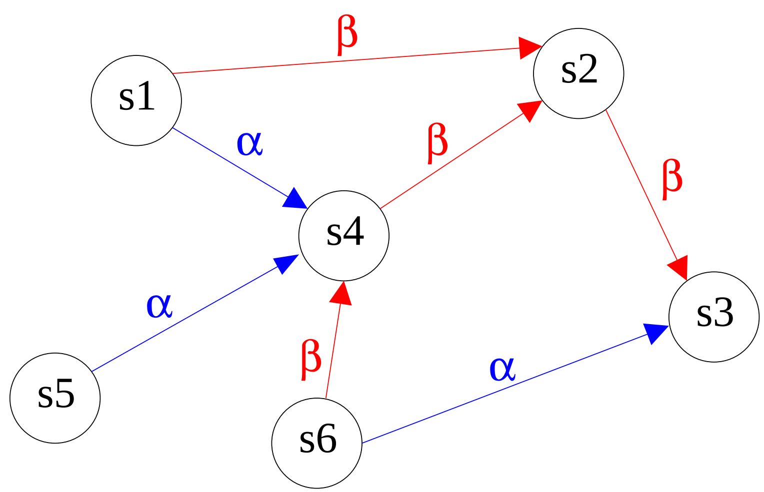
What does this mean in terms of program execution?

$\langle\cdot\rangle$**: total correctness;** $[.]$**: partial correctness**

Diamond and Box Revealed

$\langle\alpha\rangle\Phi$ **There exists an α-reachable state, such that Φ holds.**

$[\alpha]\Phi$ **Φ holds in all α-reachable states.**



What does this mean in terms of program execution?

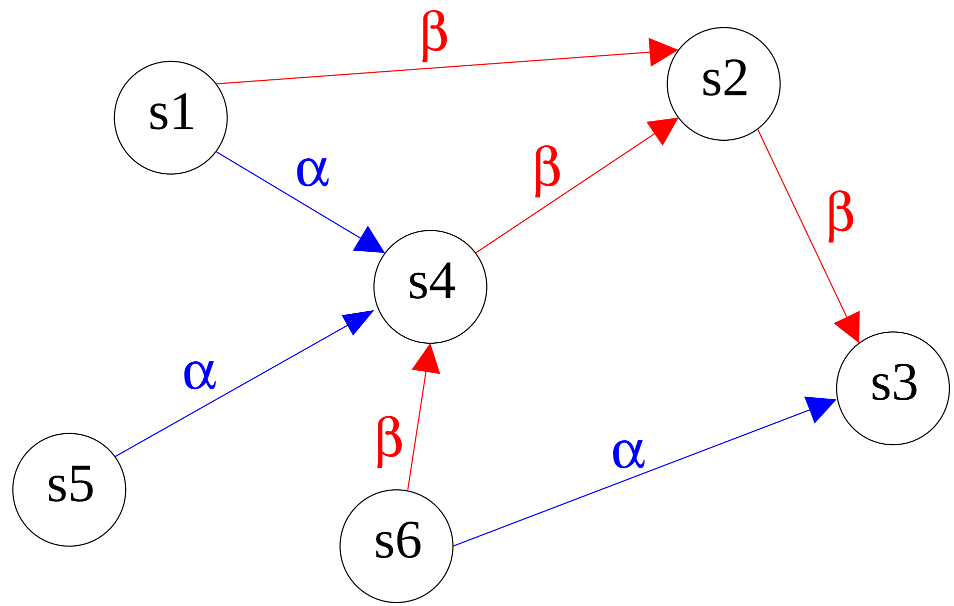
$\langle\cdot\rangle$: total correctness; $[.]$: partial correctness

<u>Duality:</u> $\langle\alpha\rangle\Phi$ iff. $\neg[\alpha]\neg\Phi$

Semantics of Dynamic Logic

Let $\mathcal{P} = \{A, B, C\}$, $\mathcal{D} = I\!N$ **and**

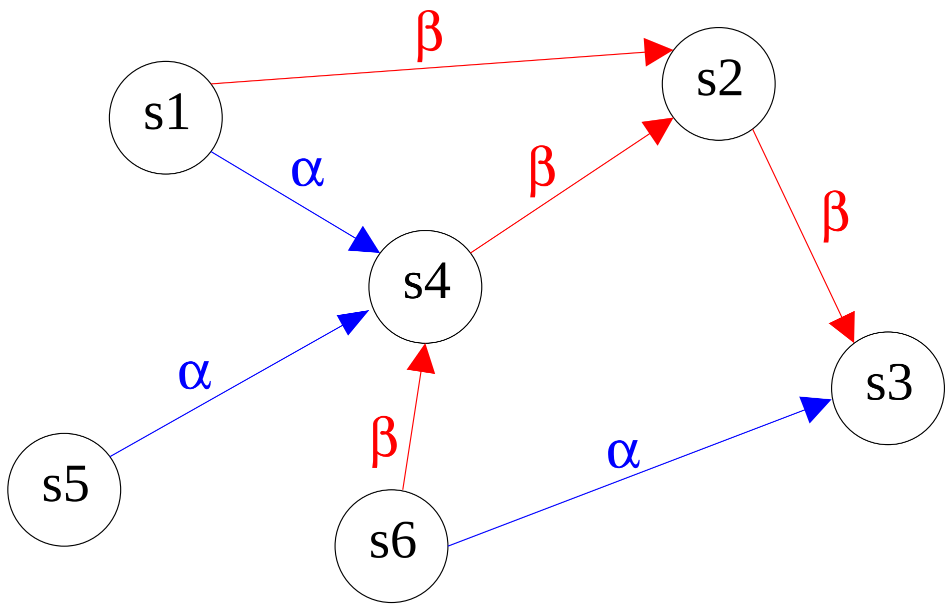
$s1 : I = \{A, B\}$, $s2 : I = \{C\}$, $s4 : I = \{A\}$



Semantics of Dynamic Logic

Let $\mathcal{P} = \{A, B, C\}$**,** $\mathcal{D} = I\!N$ **and**

$s1 : I = \{A, B\}$**,** $s2 : I = \{C\}$**,** $s4 : I = \{A\}$

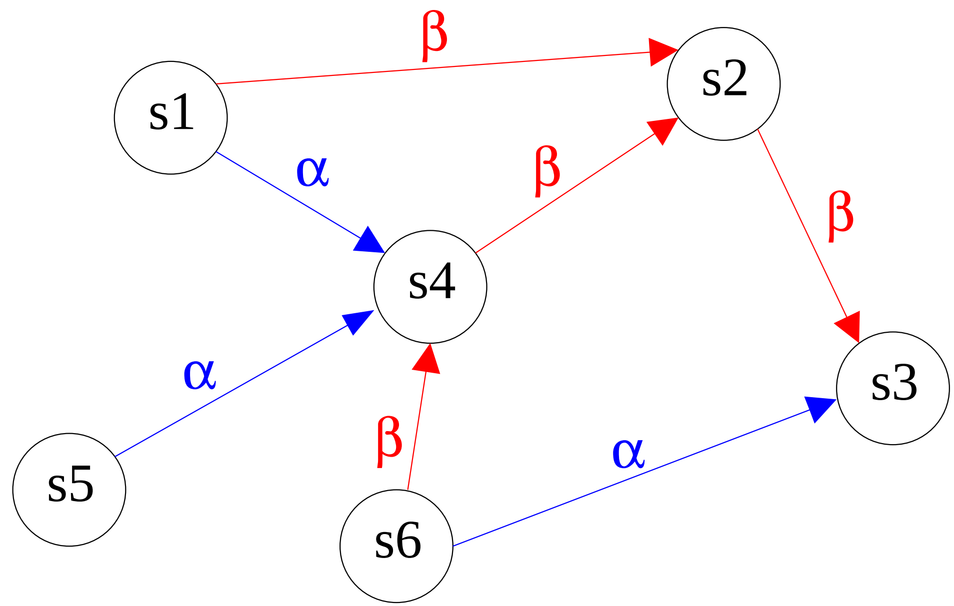


$s1 \models \langle\alpha\rangle A$**?**

Semantics of Dynamic Logic

Let $\mathcal{P} = \{A, B, C\}$**,** $\mathcal{D} = I\!N$ **and**

$s1 : I = \{A, B\}$**,** $s2 : I = \{C\}$**,** $s4 : I = \{A\}$

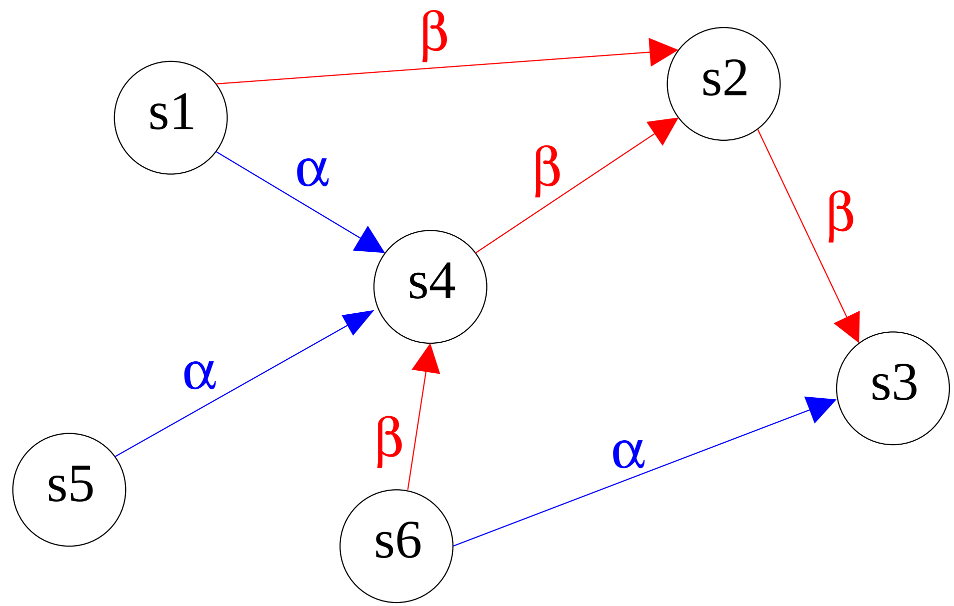


$s1 \models \langle\alpha\rangle A$ **(ok),**

Semantics of Dynamic Logic

Let $\mathcal{P} = \{A, B, C\}$, $\mathcal{D} = I\!N$ **and**

$s1 : I = \{A, B\}$, $s2 : I = \{C\}$, $s4 : I = \{A\}$

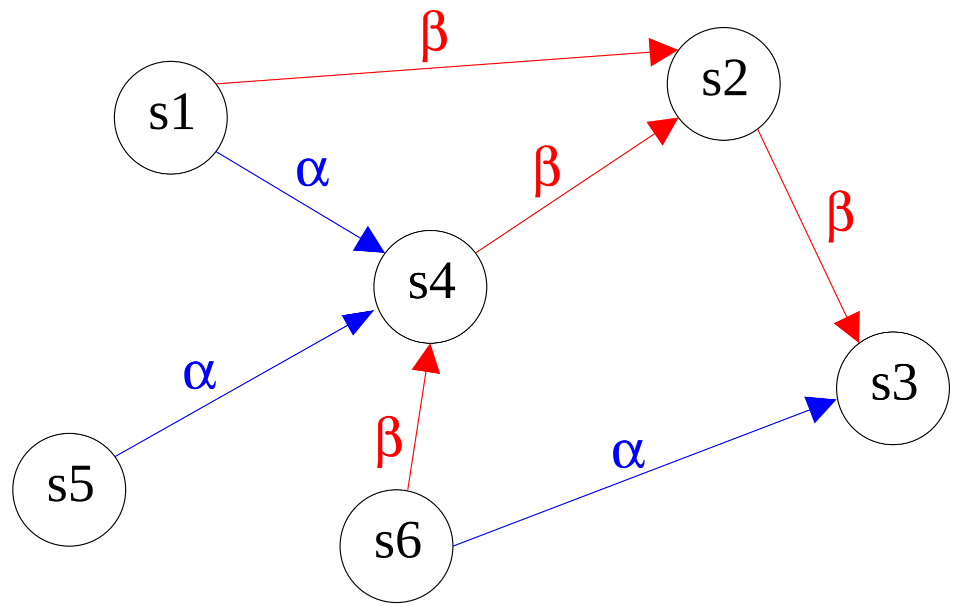


$s1 \models \langle\alpha\rangle A$ **(ok)**, $s1 \models \langle\beta\rangle A$**?**

Semantics of Dynamic Logic

Let $\mathcal{P} = \{A, B, C\}$**,** $\mathcal{D} = I\!N$ **and**

$s1 : I = \{A, B\}$**,** $s2 : I = \{C\}$**,** $s4 : I = \{A\}$

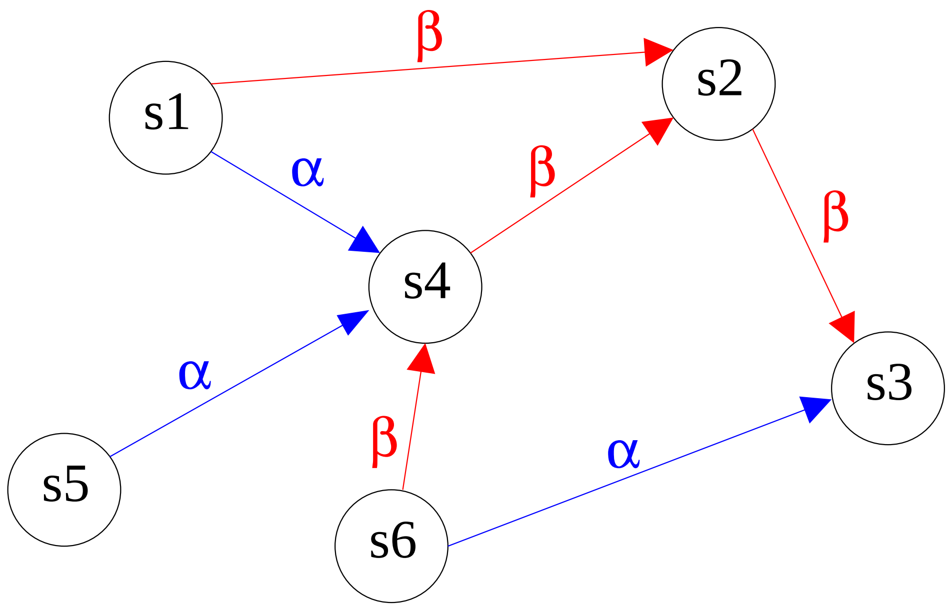


$s1 \models \langle\alpha\rangle A$ **(ok),** $s1 \models \langle\beta\rangle A$ **(—)**

Semantics of Dynamic Logic

Let $\mathcal{P} = \{A, B, C\}$, $\mathcal{D} = I\!N$ **and**

$s1 : I = \{A, B\}$, $s2 : I = \{C\}$, $s4 : I = \{A\}$



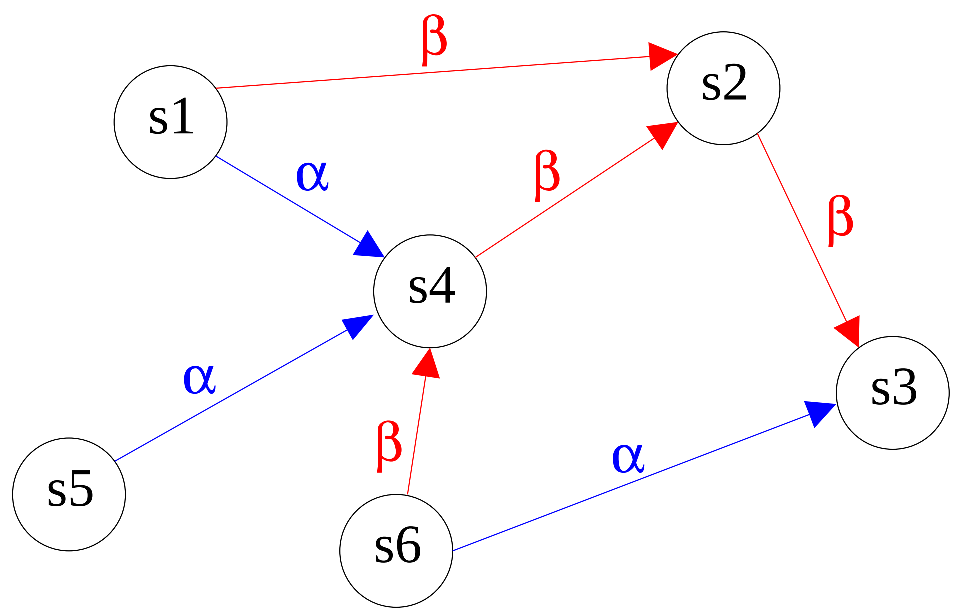
$s1 \models \langle\alpha\rangle A$ **(ok)**, $s1 \models \langle\beta\rangle A$ **(—)**

$s5 \models \langle\beta\rangle A$**?**

Semantics of Dynamic Logic

Let $\mathcal{P} = \{A, B, C\}$**,** $\mathcal{D} = I\!N$ **and**

$s1 : I = \{A, B\}$, $s2 : I = \{C\}$, $s4 : I = \{A\}$



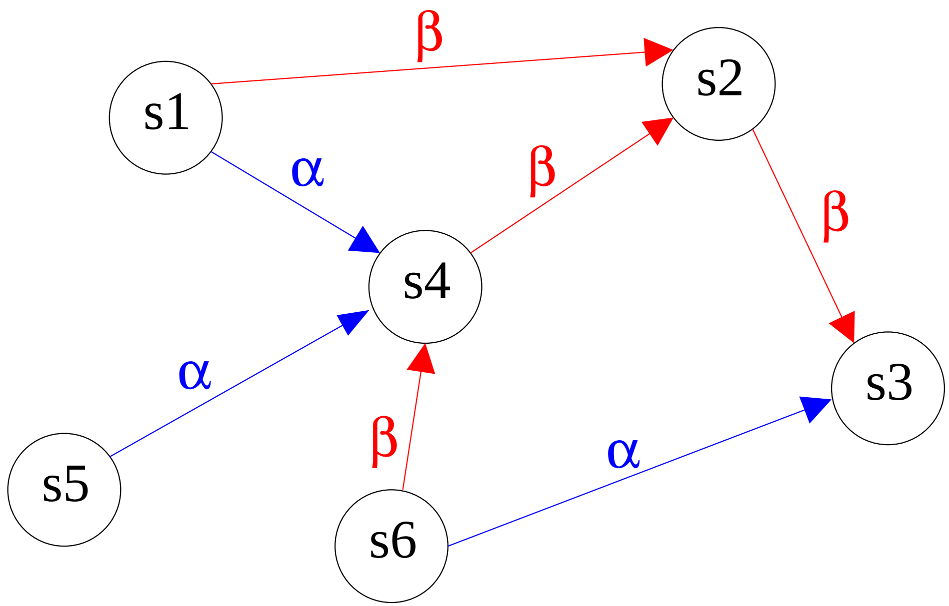
$s1 \models \langle\alpha\rangle A$ **(ok)**, $s1 \models \langle\beta\rangle A$ **(—)**

$s5 \models \langle\beta\rangle A$ **(—)**,

Semantics of Dynamic Logic

Let $\mathcal{P} = \{A, B, C\}$, $\mathcal{D} = I\!N$ **and**

$s1 : I = \{A, B\}$, $s2 : I = \{C\}$, $s4 : I = \{A\}$



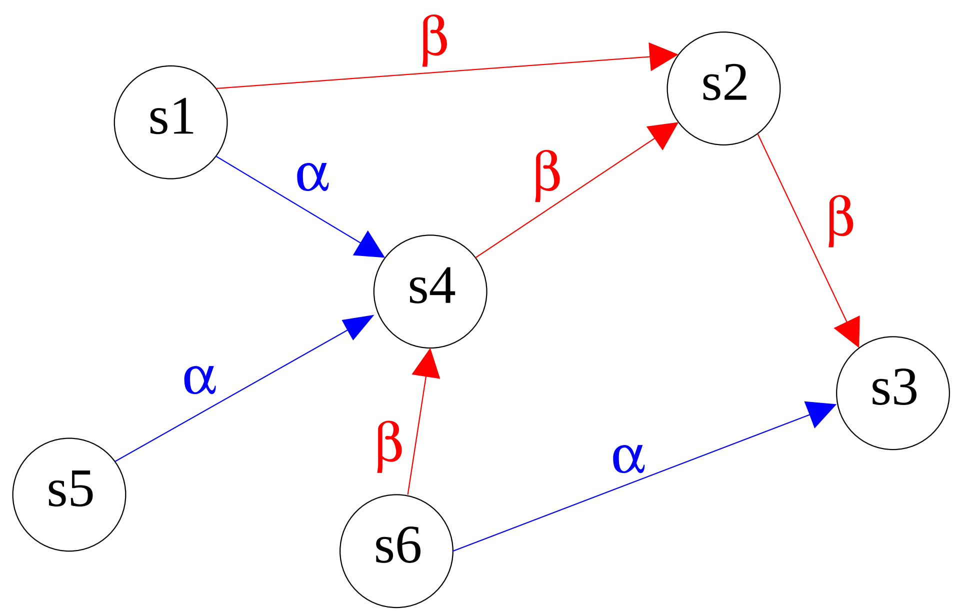
$s1 \models \langle\alpha\rangle A$ **(ok)**, $s1 \models \langle\beta\rangle A$ **(—)**

$s5 \models \langle\beta\rangle A$ **(—)**, $s5 \models [\beta]A$**?**

Semantics of Dynamic Logic

Let $\mathcal{P} = \{A, B, C\}$**,** $\mathcal{D} = I\!N$ **and**

$s1 : I = \{A, B\}$, $s2 : I = \{C\}$, $s4 : I = \{A\}$



$s1 \models \langle\alpha\rangle A$ **(ok)**, $s1 \models \langle\beta\rangle A$ **(—)**

$s5 \models \langle\beta\rangle A$ **(—)**, $s5 \models [\beta] A$ **(ok)**

A 'While'-Language with Assignments (I)

- **The atomic programs are assignments:**

  $x = t$ **(**$sort(x) = sort(t) = int$**)**

A 'While'-Language with Assignments (I)

- **The atomic programs are assignments:**

  $x = t \qquad$ **(**$sort(x) = sort(t) = int$**)**

- **Terms are arithmetical expressions (functions $+, -, *$)**

A 'While'-Language with Assignments (I)

- **The atomic programs are assignments:**

  $x = t \qquad$ **(**$sort(x) = sort(t) = int$**)**

- **Terms are arithmetical expressions (functions $+, -, *$)**

- **Conditions are built with $>$ and $>=$**

A 'While'-Language with Assignments (I)

- **The atomic programs are assignments:**

  $x = t$ **(**$sort(x) = sort(t) = int$**)**

- **Terms are arithmetical expressions (functions $+, -, *$)**
- **Conditions are built with $>$ and $>=$**

Example

```
y=1;
x=3;
while (x>0) {
  y=y*x;
  x=x-1;
}
```

A 'While'-Language with Assignments(II)

States $s = (U, I, \sigma)$

- **have all the same universe** U

A 'While'-Language with Assignments(II)

States $s = (U, I, \sigma)$

- **have all the same universe** U
- **predicate symbols are rigid**

A 'While'-Language with Assignments(II)

States $s = (U, I, \sigma)$

- **have all the same universe** U
- **predicate symbols are rigid**

Further agreement:

- **Logic variables vs. program variables:**

  Program variables cannot be quantified. Their value depends on the current state. Therefore each state contains a function

  $\sigma : ProgVar \to U$**.**

A 'While'-Language with Assignments(II)

States $s = (U, I, \sigma)$

- **have all the same universe** U
- **predicate symbols are rigid**

Further agreement:

- **Logic variables vs. program variables:**

  Program variables cannot be quantified. Their value depends on the current state. Therefore each state contains a function

  $\sigma : ProgVar \to U$.

  On the other hand, logic variables are not allowed to occur in programs and they must be bound by a quantifier.

Local Validity

There is some choice selecting the consequence relation $\models$.

The deduction theorem holds for the <u>local</u> version:

$$\Gamma \models \Phi$$

iff.

for all states g: if $g \models \Gamma$ then $g \models \Phi$

Local Validity

There is some choice selecting the consequence relation $\models$.

The deduction theorem holds for the local version:

$$\Gamma \models \Phi$$

iff.

for all states g: if $g \models \Gamma$ then $g \models \Phi$

(global version:

$$\Gamma \models \Phi$$

iff.

for all states g: $g \models \Gamma$ then for all states g: $g \models \Phi$

)

Sequent Calculus Rules

$$\textbf{IF-ELSE}\ \frac{\Gamma, b \doteq true \implies \langle\alpha\rangle\Phi, \Delta \qquad \Gamma \implies b \doteq true, \langle\beta\rangle\Phi, \Delta}{\Gamma \implies \langle \texttt{if}\ (b)\ \texttt{then}\ \alpha;\ \texttt{else}\ \beta;\rangle\Phi, \Delta}$$

Sequent Calculus Rules

IF-ELSE $$\frac{\Gamma, b \doteq true \implies \langle\alpha\rangle\Phi, \Delta \qquad \Gamma \implies b \doteq true, \langle\beta\rangle\Phi, \Delta}{\Gamma \implies \langle \texttt{if}\ (b)\ \texttt{then}\ \alpha;\ \texttt{else}\ \beta; \rangle\Phi, \Delta}$$

Assignment $$\frac{\Gamma^{x \leftarrow y}, x \doteq t \quad \vdash \quad \Phi,\ \Delta^{x \leftarrow y}}{\Gamma \quad \vdash \quad \langle x = t\rangle\Phi,\ \Delta}\ (y\ new\ variable)$$

Sequent Calculus Rules

IF-ELSE
$$\frac{\Gamma, b \doteq true \implies \langle\alpha\rangle\Phi, \Delta \qquad \Gamma \implies b \doteq true, \langle\beta\rangle\Phi, \Delta}{\Gamma \implies \langle \texttt{if}\ (b)\ \texttt{then}\ \alpha;\ \texttt{else}\ \beta;\rangle\Phi, \Delta}$$

Assignment
$$\frac{\Gamma^{x\leftarrow y}, x \doteq t \vdash \Phi,\ \Delta^{x\leftarrow y}}{\Gamma \vdash \langle x = t\rangle\Phi,\ \Delta}\ (y\ new\ variable)$$

Example

DEMO