Vorläufige Folien. Diese Folien werden nach der Vorlesung kommentarlos durch die aus der Vorlesung ersetzt.

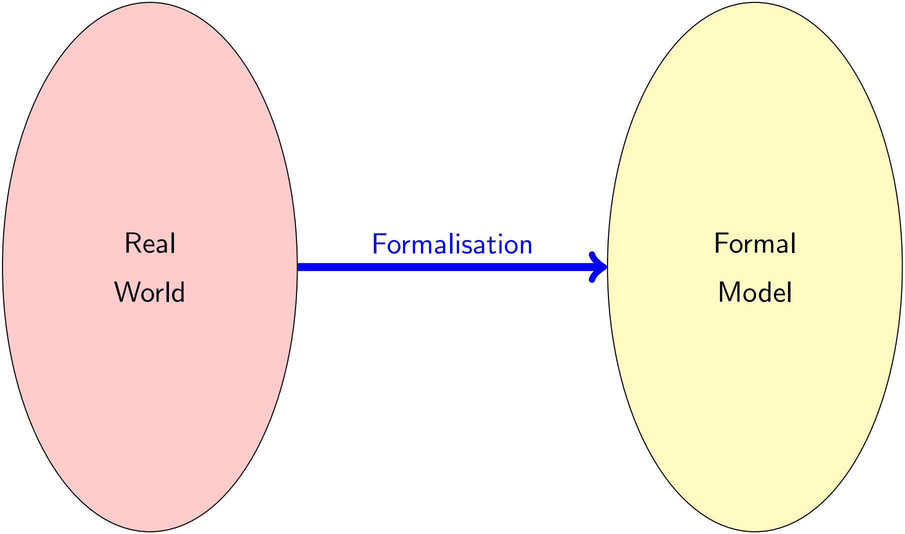
Formale Methoden im Softwareentwurf

Spezifikation mit Linearer Temporaler Logik / Specifying with Linear Temporal Logic

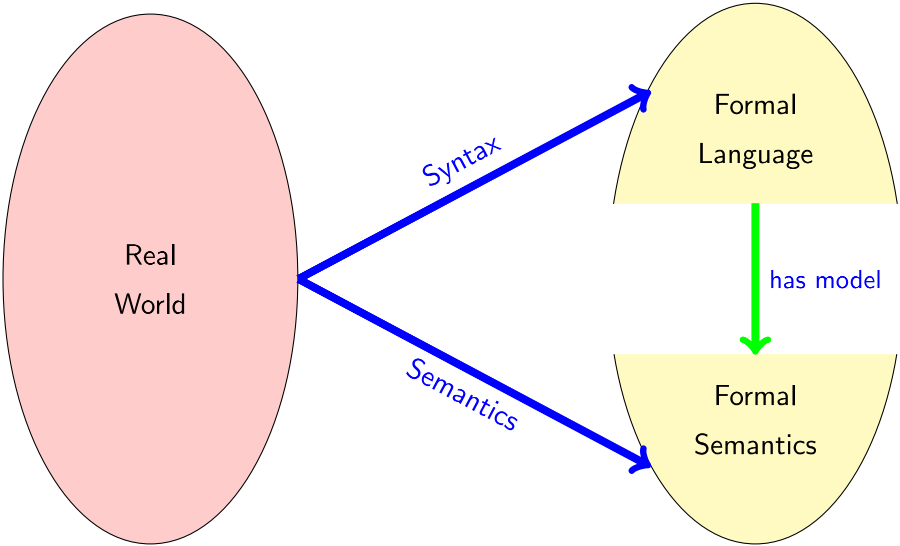
Reiner Hähnle

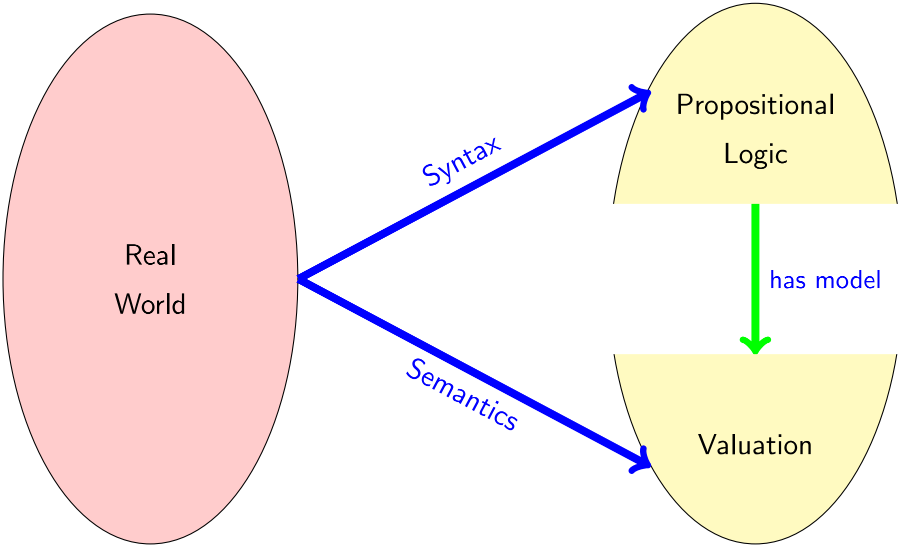
19 November 2018

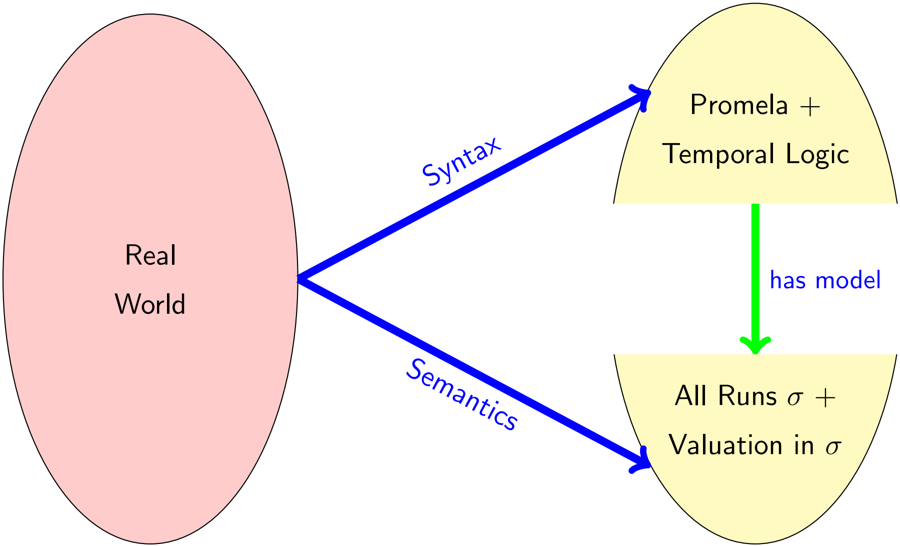
Formalisation: Syntax, Semantics, Proving

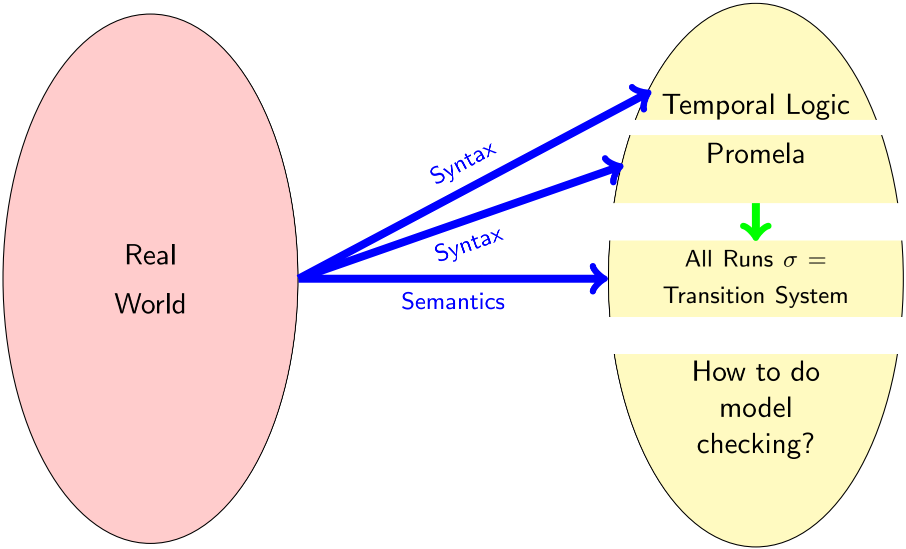


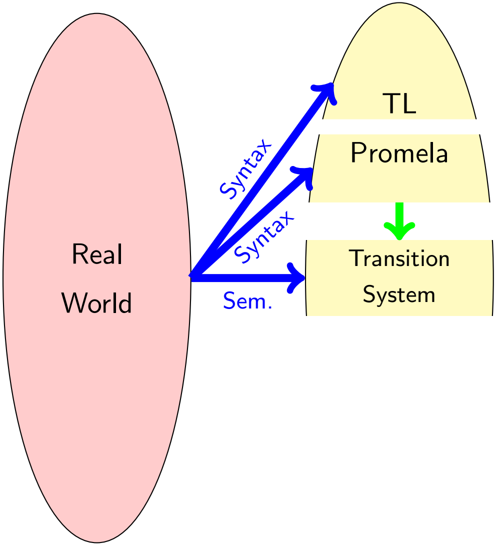
Formal Verification: Model Checking

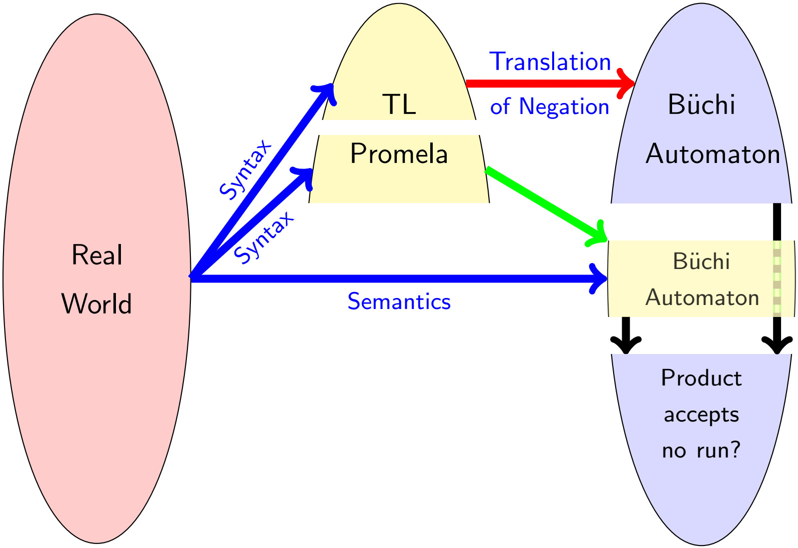



Real
World
Syntax
Semantics
Propositional
Logic
has model
Valuation

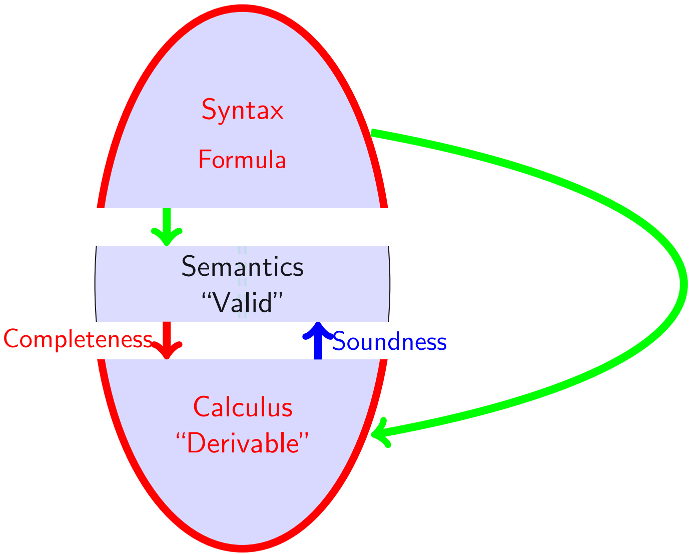

Real
World
Syntax
Semantics
Promela +
Temporal Logic
has model
All Runs σ +
Valuation in σ


Real
World
Syntax
Syntax
Semantics
Temporal Logic
Promela
All Runs $\sigma =$
Transition System
How to do
model
checking?

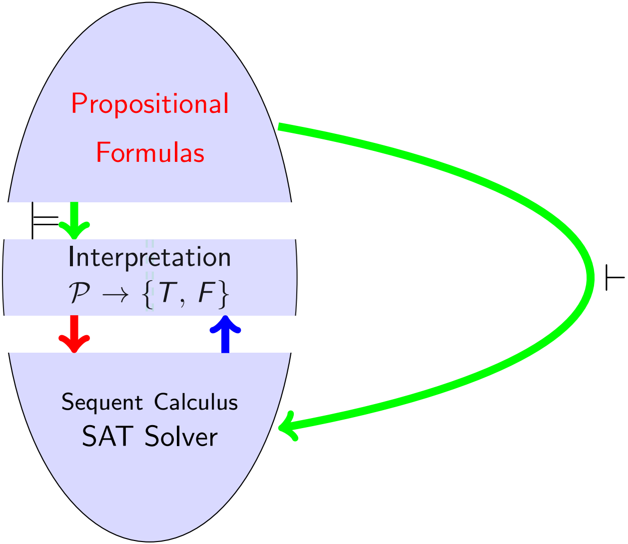

Real
World
Syntax
Syntax
Sem.
TL
Promela
Transition
System


Real
World
Syntax
Syntax
TL
Promela
Translation
of Negation
Büchi
Automaton
Semantics
Büchi
Automaton
Product
accepts
no run?

The Big Picture: Syntax, Semantics, Calculus



Simplest Case: Propositional Logic—Syntax



Syntax of Propositional Logic

Signature

A set of Propositional Variables $\mathcal{P}$ (with typical elements $p, q, r, \ldots$)

Propositional Connectives

$\mathrm{true}, \ \mathrm{false}, \ \wedge, \ \vee, \ \neg, \ \rightarrow, \ \leftrightarrow$

Set of Propositional Formulas For_0

1. Truth constants true, false and variables $\mathcal{P}$ are formulas
2. If ϕ and ψ are formulas then
$$\neg\phi, \quad (\phi \wedge \psi), \quad (\phi \vee \psi), \quad (\phi \rightarrow \psi), \quad (\phi \leftrightarrow \psi)$$
are also formulas
3. There are no other formulas (inductive definition)

Remark on Concrete Syntax

	Text book	SPIN
Negation	$\neg$	!
Conjunction	$\wedge$	&&
Disjunction	$\vee$	||
Implication	$\rightarrow$, $\supset$	->
Equivalence	$\leftrightarrow$	<->

We use mostly the textbook notation
Except for tool-specific slides, input files

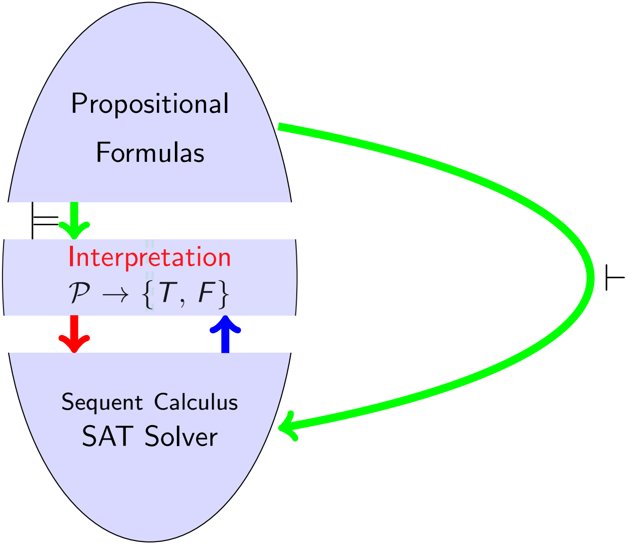
Propositional Logic Syntax: Examples

Let $\mathcal{P} = \{p, q, r\}$ be the set of propositional variables

Are the following character sequences also propositional formulas?

- $(\mathrm{true} \rightarrow p)$ ✔
- $((p(q \wedge r)) \vee p)$ ✘
- $(p \rightarrow (q \wedge))$ ✘
- $(\mathrm{false} \wedge (p \rightarrow (q \wedge r)))$ ✔

Simplest Case: Propositional Logic



Semantics of Propositional Logic

Interpretation $\mathcal{I}$

Assigns a truth value to each propositional variable

$$\mathcal{I} : \mathcal{P} \to \{T, F\}$$

Example

Let $\mathcal{P} = \{p, q\}$

$$(p \ \to \ (q \ \to \ p))$$

	p	q
$\mathcal{I}_1$	F	F
$\mathcal{I}_2$	T	F
$\vdots$	$\vdots$	$\vdots$

Semantics of Propositional Logic

Interpretation $\mathcal{I}$

Assigns a truth value to each propositional variable

$$\mathcal{I} : \mathcal{P} \to \{T, F\}$$

Valuation Function

$val_{\mathcal{I}}$: Continuation of $\mathcal{I}$ on For_0

$$val_{\mathcal{I}} : For_0 \; \to \; \{T, F\}$$

$val_{\mathcal{I}}(p_i) = \mathcal{I}(p_i)$
$val_{\mathcal{I}}(\mathrm{true}) = T$
$val_{\mathcal{I}}(\mathrm{false}) = F$

(cont'd next page)

Semantics of Propositional Logic (Cont'd)

Valuation function (Cont'd)

$$val_{\mathcal{I}}(\neg\phi) = \begin{cases} T & \text{if } val_{\mathcal{I}}(\phi) = F \\ F & otherwise \end{cases}$$

$$val_{\mathcal{I}}(\phi \wedge \psi) = \begin{cases} T & \text{if } val_{\mathcal{I}}(\phi) = T \textbf{ and } val_{\mathcal{I}}(\psi) = T \\ F & otherwise \end{cases}$$

$$val_{\mathcal{I}}(\phi \vee \psi) = \begin{cases} T & \text{if } val_{\mathcal{I}}(\phi) = T \textbf{ or } val_{\mathcal{I}}(\psi) = T \\ F & otherwise \end{cases}$$

$$val_{\mathcal{I}}(\phi \rightarrow \psi) = \begin{cases} T & \text{if } val_{\mathcal{I}}(\phi) = F \textbf{ or } val_{\mathcal{I}}(\psi) = T \\ F & otherwise \end{cases}$$

$$val_{\mathcal{I}}(\phi \leftrightarrow \psi) = \begin{cases} T & \text{if } val_{\mathcal{I}}(\phi) = val_{\mathcal{I}}(\psi) \\ F & otherwise \end{cases}$$

Valuation Examples

Example

Let $\mathcal{P} = \{p, q\}$

$$(p \rightarrow (q \rightarrow p))$$

	p	q
$\mathcal{I}_1$	F	F
$\mathcal{I}_2$	T	F
...		

How to evaluate $(p \rightarrow (q \rightarrow p))$ in $\mathcal{I}_2$?

Valuation in $\mathcal{I}_2$

$val_{\mathcal{I}_2}(\ q \rightarrow p\ ) = T$ because $val_{\mathcal{I}_2}(q) = F$ or $val_{\mathcal{I}_2}(p) = T$

$val_{\mathcal{I}_2}(\ p \rightarrow (q \rightarrow p)\ ) = T$

because $val_{\mathcal{I}_2}(p) = F$ or $val_{\mathcal{I}_2}(\ q \rightarrow p\ ) = T$

Semantic Notions of Propositional Logic

Let $\phi \in For_0$, $\Gamma \subseteq For_0$

Definition (Satisfying Interpretation, Consequence Relation)

$\mathcal{I}$ satisfies ϕ (write: $\mathcal{I} \models \phi$) iff $val_{\mathcal{I}}(\phi) = T$

ϕ follows from Γ (write: $\Gamma \models \phi$) iff for all interpretations $\mathcal{I}$:

$$\text{If } \mathcal{I} \models \psi \text{ for all } \psi \in \Gamma \text{ then also } \mathcal{I} \models \phi$$
$$\text{iff}$$
$$\{\mathcal{I} \mid \mathcal{I} \models \psi \text{ for all } \psi \in \Gamma\} \subseteq \{\mathcal{I} \mid \mathcal{I} \models \phi\}$$

Definition (Satisfiability, Validity)

A formula is satisfiable if it is satisfied by some interpretation.
If every interpretation satisfies ϕ (write: $\models \phi$) then ϕ is called valid.

Semantics of Propositional Logic: Examples

Formula (same as before)

$$p \rightarrow (q \rightarrow p)$$

Is this formula valid?

$$\models p \rightarrow (q \rightarrow p)\ ?$$

Yes. How to prove?

Semantics of Propositional Logic: Examples

$$p \wedge ((\neg p) \vee q)$$

Satisfiable?	✔
Satisfying Interpretation?	$\mathcal{I}(p) = T, \mathcal{I}(q) = T$
Other Satisfying Interpretations?	✘
Therefore, also not valid!	

$$p \wedge ((\neg p) \vee q) \models q \vee r$$

Does it hold? Yes. Why?

An Easy Exercise in Formalisation

Knights & Knaves

In old times there existed an island whose inhabitants were either knights or knaves. A knight always tells the truth, while a knave always lies. Hedwig and Katrin lived on that Island. A historian found in the archives the following statements:

Hedwig: I am a knave if and only if Katrin is a knave.

Katrin: We are of different kind.

(after Raymond Smullyan, Knight and Knaves)

Who is who?

Try to formalise the puzzle in propositional logic

A Harder Exercise in Formalisation

```
// Program P
byte n;
active proctype [2] p() {
  n = 0;
  n = n + 1
}
```

How can we formalize the PROMELA program P in propositional logic?

P is represented by a propositional formula ϕ_{P} provided that an interpretation $\mathcal{I}$ satisfies ϕ_{P} iff $\mathcal{I}$ describes a possible state of P

A Harder Exercise in Formalisation

```
1 // Program P
2 byte n;
3 active proctype [2] p() {
4   n = 0;
5   n = n + 1
6 }
```

$\mathcal{P}$: $N_0, N_1, N_2, \ldots, N_7$ 8-bit representation of **byte** `n`
PCi_j counter when next instruction of process i at line $j \in \{3, 4, 5\}$

Which interpretations do we need to "exclude"?

- The variable `n` is represented by eight bits, all values possible
- A process cannot be at two positions at the same time
- If neither process 0 nor process 1 are at position 5, then `n` is zero
- ...

$$\phi_{\mathrm{P}} := \begin{pmatrix} ((PC0_3 \wedge \neg PC0_4 \wedge \neg PC0_5) \vee \cdots) \wedge \\ ((\neg PC0_5 \wedge \neg PC1_5) \implies (\neg N_0 \wedge \cdots \wedge \neg N_7)) \wedge \cdots \end{pmatrix}$$

Is Propositional Logic Enough?

Can design for a program P a formula Φ_P describing all reachable states

For a given property Ψ the consequence relation

$$\Phi_p \models \Psi$$

holds when Ψ is true in every state reachable in every run of P

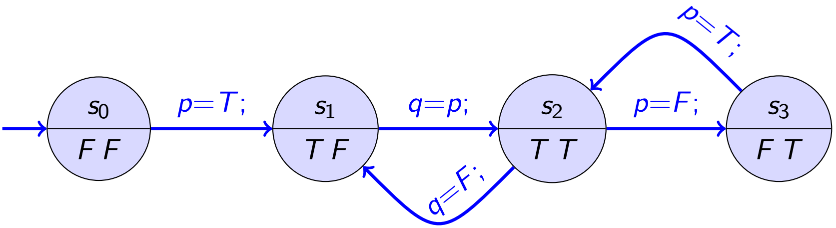
How to Express Properties Involving State Changes?

In every run of a program P

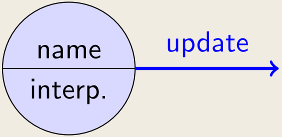
- n will become greater than 0 eventually?
- n changes its value infinitely often
- ...

$\Rightarrow$ Need a more expressive logic: Linear Temporal Logic

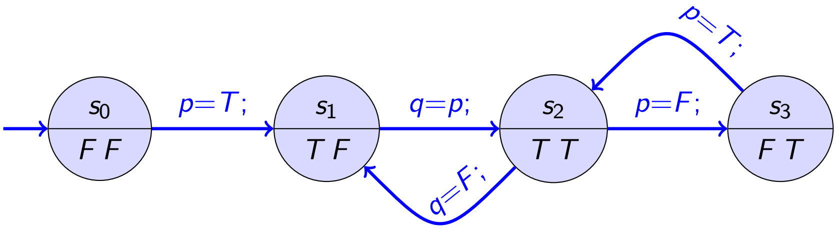
Semantics: Transition systems (Kripke Structures)



Notation



Semantics: Transition systems (Kripke Structures)

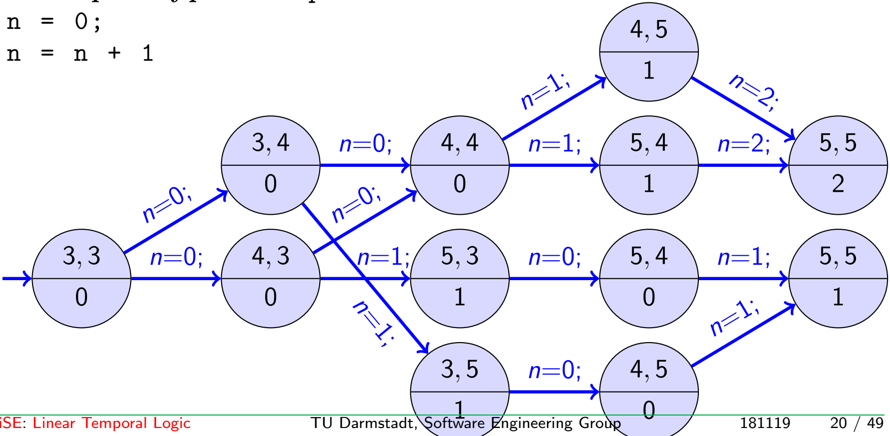


- Each (program) state s_j has its own propositional interpretation $\mathcal{I}_j$
  - Convention: list values of variables in ascending lexicographic order
- Computations, or runs, are infinite paths through states
  - "infinite" not a restriction: let finite run be "stuck" at final state
- In general, infinitely many different runs possible
- How to express (for example) that p changes its value infinitely often in each run?

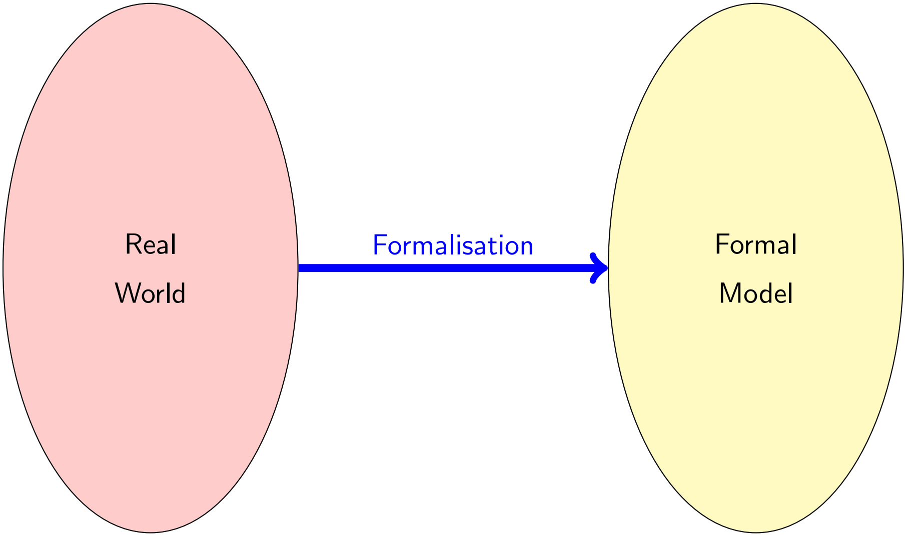
PROMELA Programs and Transition Systems

Runs of PROMELA program ≈ runs of suitable transition system

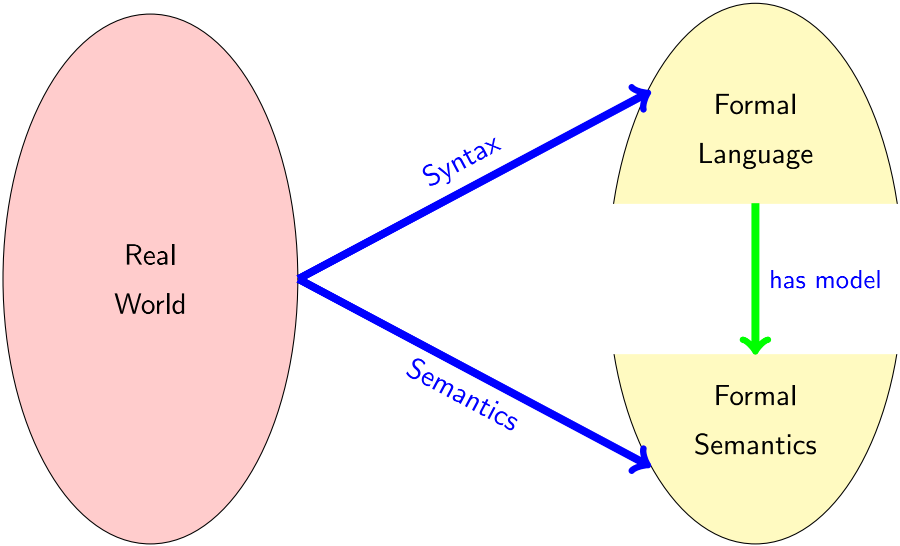
```
// Program P
byte n;
active proctype [2] p() {
  n = 0;
  n = n + 1
}
```

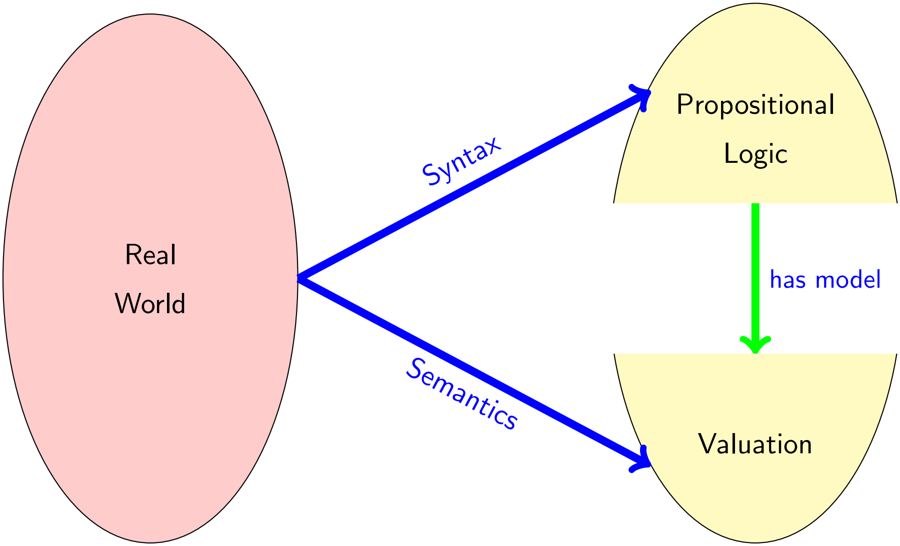


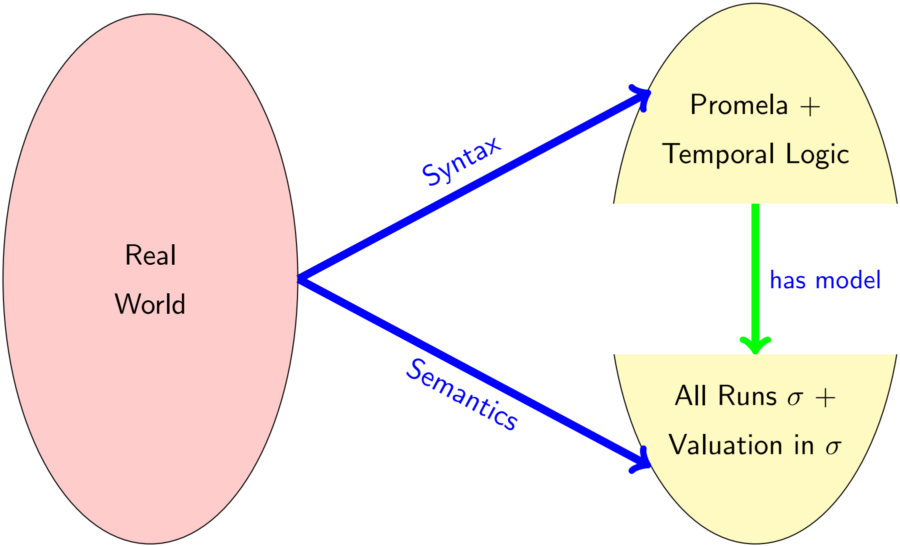
Formalisation: Syntax, Semantics, Proving

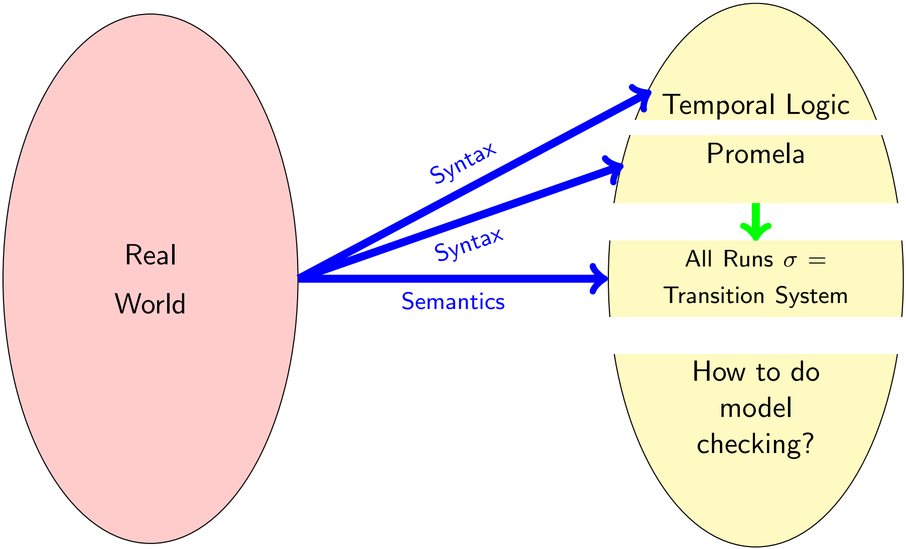


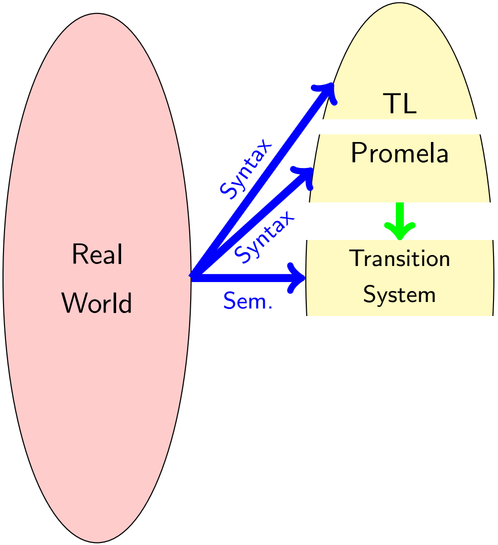
Formal Verification: Model Checking

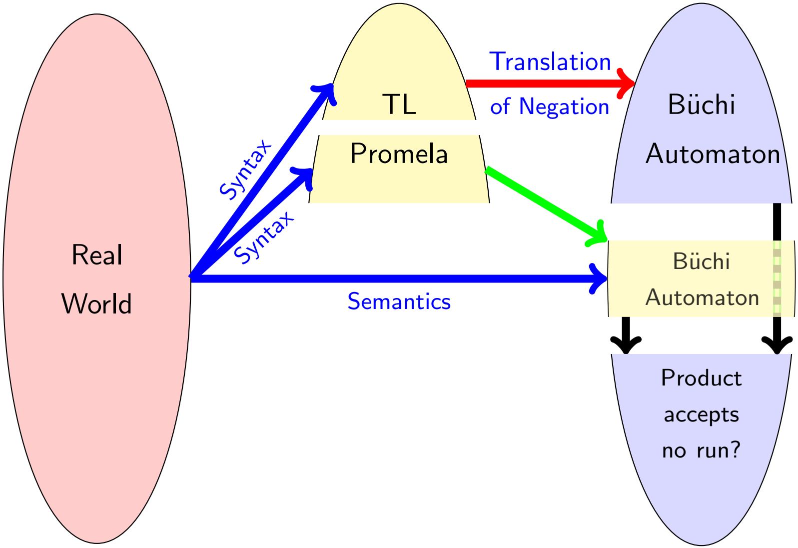



Real
World
Syntax
Semantics
Propositional
Logic
has model
Valuation


Real
World
Syntax
Semantics
Promela +
Temporal Logic
has model
All Runs σ +
Valuation in σ


Real
World
Syntax
Syntax
Semantics
Temporal Logic
Promela
All Runs σ =
Transition System
How to do
model
checking?


Real
World
Syntax
Syntax
Sem.
TL
Promela
Transition
System


Real
World
Syntax
Syntax
Semantics
TL
Promela
Translation
of Negation
Büchi
Automaton
Büchi
Automaton
Product
accepts
no run?

Temporal Logic—Syntax

An extension of propositional logic that
allows to specify properties of sets of runs

Syntax

Based on propositional signature and syntax

Extension with three connectives:

Always If ϕ is a formula then so is $\Box\phi$

Sometimes If ϕ is a formula then so is $\Diamond\phi$

Until If ϕ and ψ are formulas then so is $\phi\mathcal{U}\psi$

Concrete Syntax

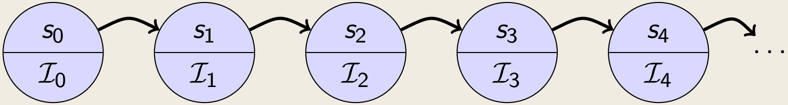
	text book	SPIN
Always	$\Box$	[]
Sometimes	$\Diamond$	<>
Until	$\mathcal{U}$	U

Temporal Logic—Semantics

Need to generalize semantics of propositional logic

- Propositional formula evaluated relative to one interpretation
- Temporal formula evaluated relative to sequence of interpretations

A run σ of a transition system is an infinite chain of states



$\mathcal{I}_j$ propositional interpretation of variables in j-th state
Write run more compactly $s_0\, s_1\, s_2\, s_3 \ldots$

If $\sigma = s_0\, s_1 \cdots$, then $\sigma|_i$ denotes the suffix $s_i\, s_{i+1} \cdots$ of σ

Temporal Logic—Semantics (Cont'd)

Valuation of temporal formula relative to run: infinite sequence of states

Definition (Validity Relation)

Validity of temporal formula depends on runs $\sigma = s_0\, s_1 \ldots$

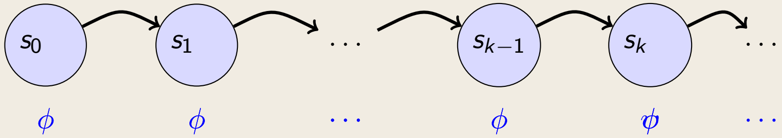
$\sigma \models p$	iff	$\mathcal{I}_0(p) = T$, for $p \in \mathcal{P}$
$\sigma \models \neg\phi$	iff	not $\sigma \models \phi$ (write $\sigma \not\models \phi$)
$\sigma \models \phi \land \psi$	iff	$\sigma \models \phi$ and $\sigma \models \psi$
$\sigma \models \phi \lor \psi$	iff	$\sigma \models \phi$ or $\sigma \models \psi$
$\sigma \models \phi \to \psi$	iff	$\sigma \not\models \phi$ or $\sigma \models \psi$

Propositional formulas evaluated in interpretation of initial state of σ

Temporal connectives?

Temporal Logic—Semantics (Cont'd)

Run σ



Definition (Validity Relation for Temporal Connectives)

Given a run $\sigma = s_0\, s_1 \cdots s_{k-1}\, s_k \cdots$

$\sigma \models \Box\phi$ iff $\sigma|_k \models \phi$ for all $k \geq 0$

$\sigma \models \Diamond\phi$ iff $\sigma|_k \models \phi$ for some $k \geq 0$

$\sigma \models \phi\mathcal{U}\psi$ iff $\sigma|_k \models \psi$ for some $k \geq 0$, and $\sigma|_j \models \phi$ for all $0 \leq j < k$
(if $k = 0$ then ϕ needs never hold)

*

Safety and Liveness Properties

Safety Properties

- Always-formulas called safety property:
  "something bad never happens"
- Let `mutex` ("mutual exclusion") be a variable that is true when two processes do not access a critical resource at the same time
- $\Box\,\texttt{mutex}$ expresses that simultaneous access never happens

Liveness Properties

- Sometimes-formulas called liveness property:
  "something good happens eventually"
- Let `s` be a variable that is true when a process delivers a service
- $\Diamond\,\texttt{s}$ expresses that this service is eventually provided

A Complex Property

What does this mean?Infinitely Often

$$\sigma \models \Box\Diamond\phi$$

"During run σ the formula ϕ becomes true infinitely often"

Validity of Temporal Logic

Definition (Validity)

ϕ is valid, write $\models \phi$, iff ϕ is valid in all runs $\sigma = s_0\, s_1 \cdots$.

Recall that each run $s_0\, s_1 \cdots$ essentially is an infinite sequence of interpretations $\mathcal{I}_0\, \mathcal{I}_1 \cdots$

Representation of Runs

Can represent a set of runs as a sequence of propositional formulas:

- $\phi_0\, \phi_1 \cdots$ represents all runs $s_0\, s_1 \cdots$ such that $\mathcal{I}_j \models \phi_j$ for $j \geq 0$

Semantics of Temporal Logic: Examples

$$\Diamond\Box\phi$$

Valid?

No, there is a run where it is not valid:
$(\neg\phi\,\neg\phi\,\neg\phi\ \cdots)$

Valid in some run?

Yes, for example: $(\neg\phi\,\phi\,\phi\ \cdots)$

$$\Box\phi \rightarrow \phi \qquad (\neg\Box\phi) \leftrightarrow (\Diamond\neg\phi) \qquad \Diamond\phi \leftrightarrow (\mathrm{true}\,\mathcal{U}\phi)$$

All are valid! (proof is exercise)

- $\Box$ is reflexive
- $\Box$ and $\Diamond$ are dual connectives
- $\Box$ and $\Diamond$ can be expressed with $\mathcal{U}$ only

Semantics of Temporal Logic: More Examples

$$(\phi \; \mathcal{U} \; \psi) \rightarrow (\phi \; \mathcal{U} \; \Box\psi)$$

Valid?

No, there is a run where it is not valid:
$(\psi \, \neg\psi \, \neg\psi \; \cdots)$

Valid in some run?

Yes, for example: $(\psi \, \psi \; \cdots)$ or $(\mathrm{false}, \mathrm{false} \; \cdots)$

$$\Box\psi \rightarrow (\phi \; \mathcal{U} \; \psi)$$

Valid! (proof is exercise)

Transition Systems: Formal Definition

Definition (Transition System)

A transition system $\mathcal{T} = (S, Ini, \delta, \mathcal{I})$ is composed of:

- set of states S
- set $\emptyset \neq Ini \subseteq S$ of initial states
- transition relation $\delta \subseteq S \times S$
- labeling $\mathcal{I}$ of each state $s \in S$ with a propositional interpretation $\mathcal{I}_s$

Definition (Run of Transition System)

A run of $\mathcal{T}$ is a sequence of states $\sigma = s_0\, s_1 \cdots$ such that $s_0 \in Ini$ and for all i is $(s_i, s_{i+1}) \in \delta$.

Temporal Logic—Semantics (Cont'd)

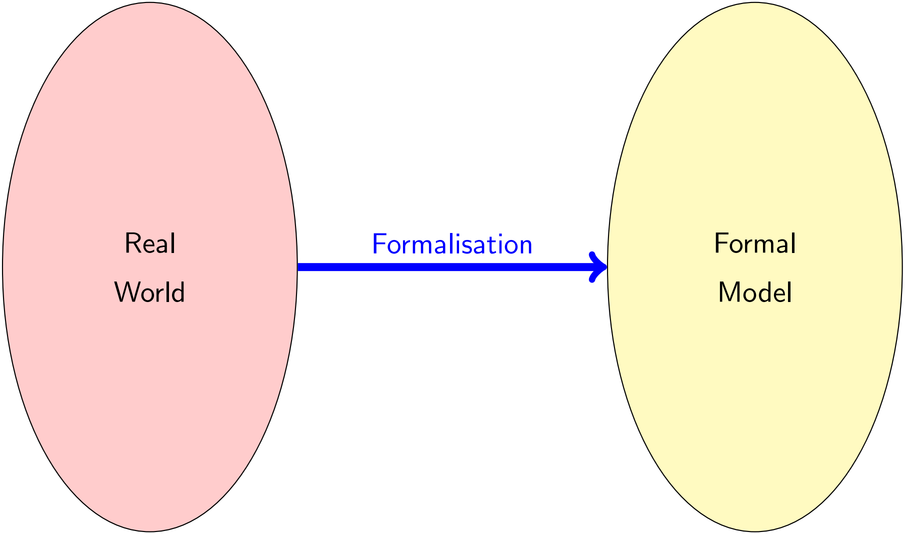
Extension of validity of temporal formulas to transition systems:

Definition (Validity Relation)

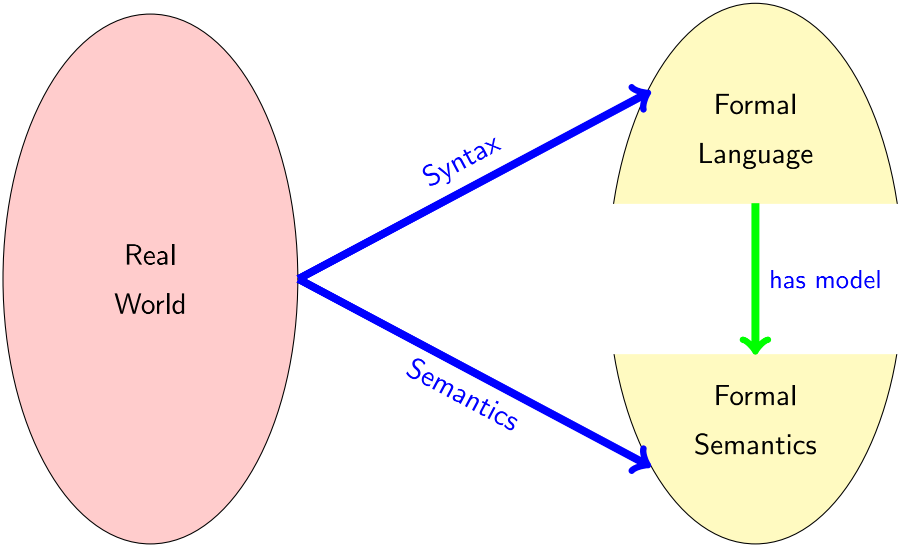
Given a transition system $\mathcal{T} = (S, \mathit{Ini}, \delta, \mathcal{I})$, a temporal formula ϕ is valid in $\mathcal{T}$ (write $\mathcal{T} \models \phi$) iff $\sigma \models \phi$ for all runs σ of $\mathcal{T}$.

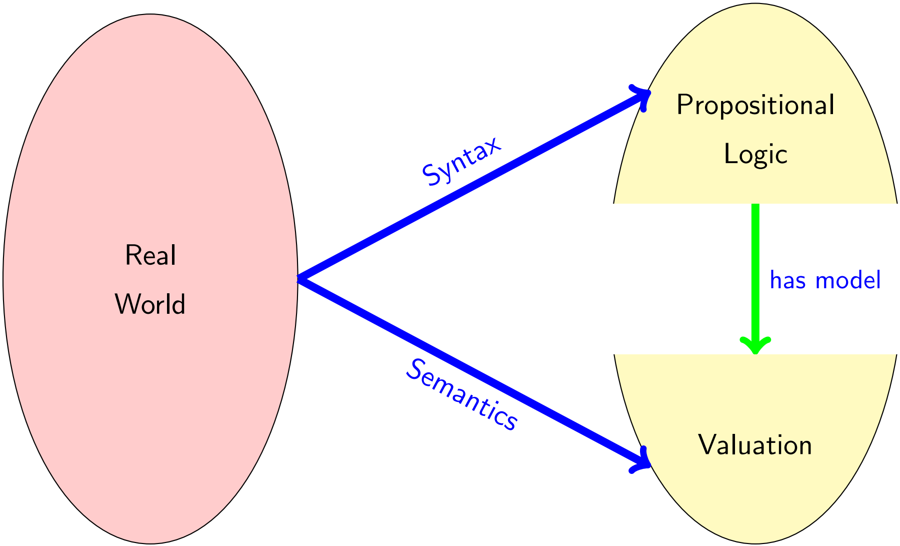
We could stop here, but transition systems hard to automate efficiently

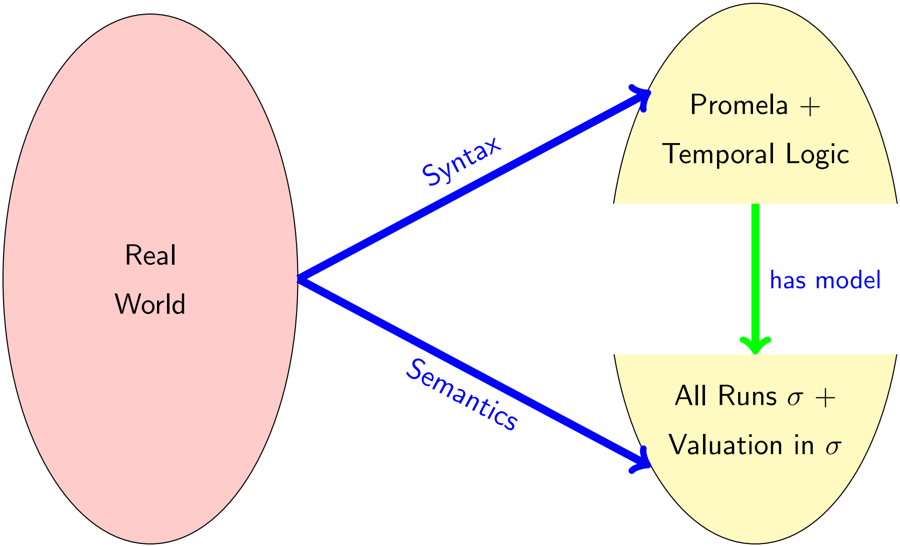
Formalisation: Syntax, Semantics, Proving

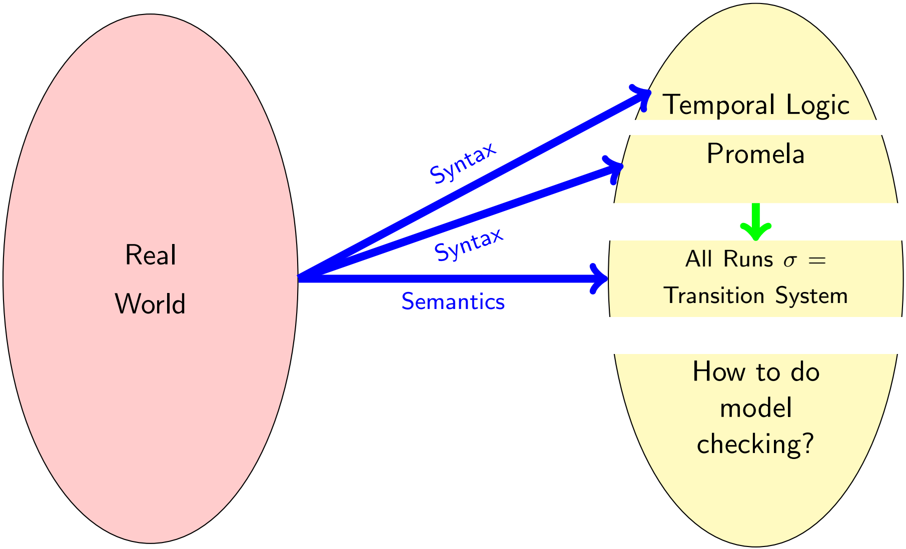


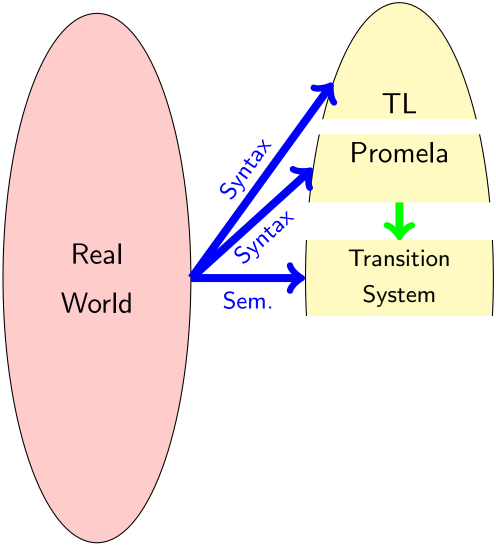
Formal Verification: Model Checking

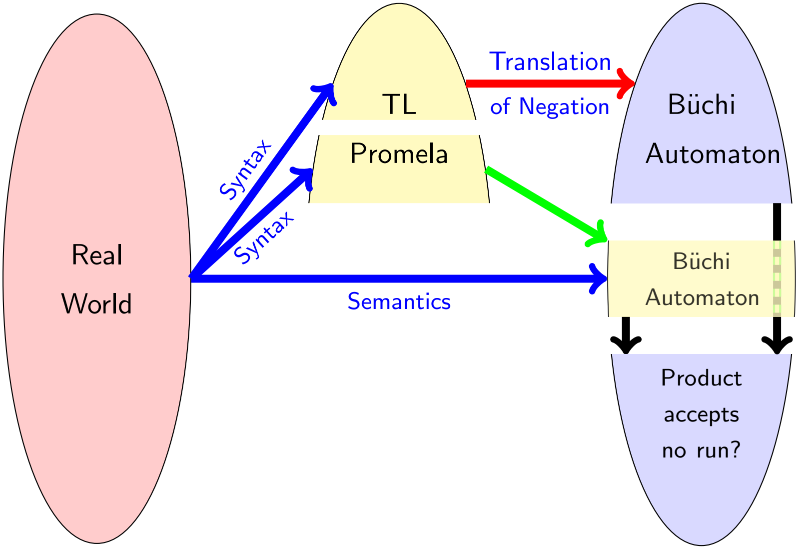



Real
World
Syntax
Semantics
Propositional
Logic
has model
Valuation


Real
World
Syntax
Semantics
Promela +
Temporal Logic
has model
All Runs σ +
Valuation in σ


Temporal Logic
Promela
Syntax
Syntax
Real
World
All Runs $\sigma =$
Transition System
Semantics
How to do
model
checking?


Real
World
Syntax
Syntax
Sem.
TL
Promela
Transition
System


Translation
of Negation
TL
Promela
Büchi
Automaton
Syntax
Syntax
Real
World
Semantics
Büchi
Automaton
Product
accepts
no run?

Formal ω-Languages

Given a finite alphabet (vocabulary) Σ

An ω-word $w \in \Sigma^{*\omega}$ is a n infinite sequence

$$w = a_o \cdots a_{nk} \cdots$$

with $a_i \in \Sigma, i \in \{0, \ldots, n\}\mathbb{N}$

$\mathcal{L}^\omega \subseteq \Sigma^{*\omega}$ is called a n ω-language over Σ

Büchi Automaton

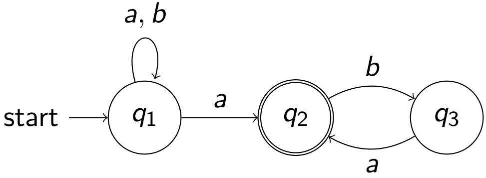
Definition (Büchi Automaton)

A (non-deterministic) Büchi automaton over an alphabet Σ consists of a

- finite, non-empty set of locations (or states) Q
- a non-empty set of initial/start locations $I \subseteq Q$
- a set of accepting/final locations $F = \{F_1, \ldots, F_n\} \subseteq Q$
- a transition relation $\delta \subseteq Q \times \Sigma \times Q$

Example

$\Sigma = \{a, b\}, Q = \{q_1, q_2, q_3\}, I = \{q_1\}, F = \{q_2\}$



Büchi Automaton—Acceptance

Definition (Run and Accepted Run)

An infinite word $w = a_o \cdots a_k \cdots \in \Sigma^\omega$ is a run of a Büchi automaton if

$$q_{i+1} \in \delta(q_i, a_i)$$

for all $i \geq 0$ and some initial location $q_0 \in I$.

A Büchi automaton accepts a run $w \in \Sigma^\omega$, if some accepting location $f \in F$ is infinitely often visited during w.

Let $\mathcal{B} = (Q, I, F, \delta)$ be a Büchi automaton, then

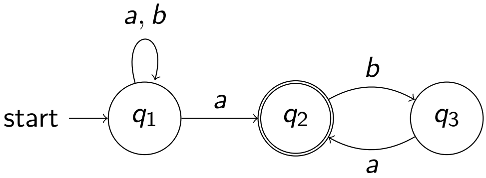
$$\mathcal{L}^\omega(\mathcal{B}) = \{w \in \Sigma^\omega | w \in \Sigma^\omega \text{ is an accepted run of } \mathcal{B}\}$$

denotes the ω-language recognized by $\mathcal{B}$

An ω-language for which an accepting Büchi automaton exists is called ω-regular language

Example, ω-Regular Expression

Which language is accepted by the following Büchi automaton?



Solution: $(a+b)^*(ab)^\omega$ [NB: $(ab)^\omega = a(ba)^\omega$]

ω-regular expressions like standard regular expression

ab a **then** b

$a+b$ a **or** b

a^* arbitrarily, but finitely often a

new: a^ω infinitely often a

Decidability, Closure Properties

Many properties for regular finite automata hold also for Büchi automata

Theorem (Decidability)

It is decidable whether the accepted language $\mathcal{L}^\omega(\mathcal{B})$ of a Büchi automaton $\mathcal{B}$ is empty.

Theorem (Closure properties)

The set of ω-regular languages is closed with respect to intersection, union and complement:

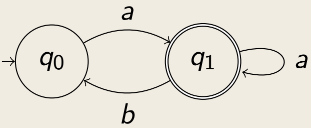
- *if $\mathcal{L}_1, \mathcal{L}_2$ are ω-regular then $\mathcal{L}_1 \cap \mathcal{L}_2$ and $\mathcal{L}_1 \cup \mathcal{L}_2$ are ω-regular*
- *$\mathcal{L}$ is ω-regular then $\Sigma^\omega \backslash \mathcal{L}$ is ω-regular*

But in contrast to regular finite automata

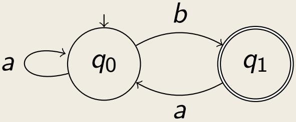
Non-deterministic Büchi automata are strictly more expressive than deterministic ones (latter cannot accept all ω-regular expressions)

Büchi Automata—More Examples

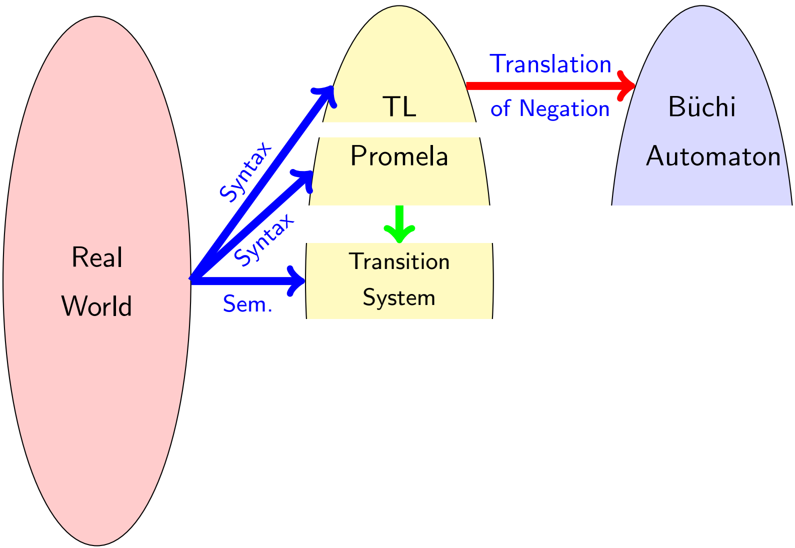
Language? $a(a + ba)^\omega$



Language? $(a^* ba)^\omega$



Formal Verification: Model Checking



Linear Temporal Logic and Büchi Automata

LTL and Büchi Automata are connected

Recall

Definition (Validity Relation)

Given a transition system $\mathcal{T} = (S, Ini, \delta, \mathcal{I})$, a temporal formula ϕ is valid in $\mathcal{T}$ (write $\mathcal{T} \models \phi$) iff $\sigma \models \phi$ for all runs σ of $\mathcal{T}$.

A run of the transition system is an infinite sequence of interpretations I

Intended Connection

Given an LTL formula ϕ:

Construct a Büchi automaton accepting exactly those runs (infinite sequences of interpretations) that satisfy ϕ

Encoding an LTL Formula as a Büchi Automaton

$\mathcal{P}$ set of propositional variables, e.g., $\mathcal{P} = \{r, s\}$

Alphabet Σ of Büchi automaton

A state transition of Büchi automaton must represent an interpretation

Let Σ be set of all interpretations over $\mathcal{P}$, i.e., $\Sigma = 2^{\mathcal{P}}$

Example

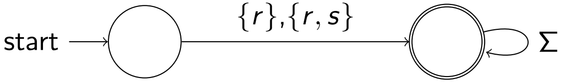
$\Sigma = \big\{\emptyset, \{r\}, \{s\}, \{r, s\}\big\}$

$$I_\emptyset(r) = F, I_\emptyset(s) = F, I_{\{r\}}(r) = T, I_{\{r\}}(s) = F, \ldots$$

Büchi Automaton for LTL Formula By Example

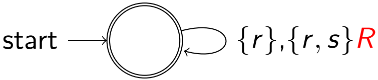
Example (Büchi automaton for formula r over $\mathcal{P} = \{r, s\}$)

A Büchi automaton $\mathcal{B}$ accepting exactly those runs σ satisfying r



In $s_0 \in \sigma$ at least r must hold, the rest is arbitrary

Example (Büchi automaton for formula $\Box r$ over $\mathcal{P} = \{r, s\}$)

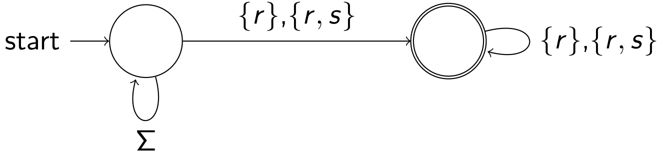


$(R := \{I | I \in \Sigma, r \in I\})$

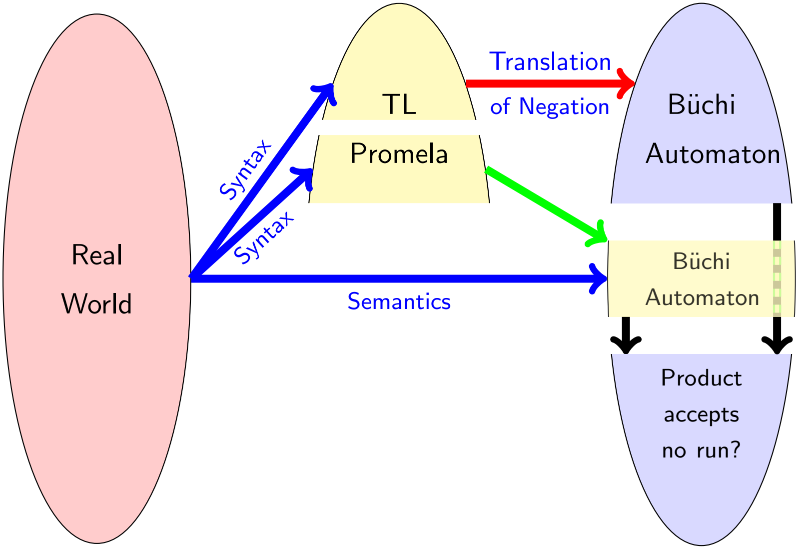
In any $s \in \sigma$ at least r must hold

Büchi Automaton for LTL Formula By Example

Example (Büchi automaton for formula $\Diamond\Box r$ over $\mathcal{P} = \{r, s\}$)



Formal Verification: Model Checking



Model Checking

Check whether an LTL formula is valid in all runs of a transition system

Given a transition system $\mathcal{T}$ (e.g., derived from a PROMELA program)

Verification task: is the LTL formula ϕ satisfied in all runs of $\mathcal{T}$, i.e.,

$$\mathcal{T} \models \phi \quad ?$$

Temporal model checking with SPIN: Topic of next lecture

Today: Basic principle behind SPIN model checking

Next Time: LTS to Büchi

LTS are Büchi Automata:

- Transitions labeled with $p := 1$ ($p := 0$) get label p ($\neg p$)
- Start state is initial state
- States corresponding to final process statements and locations with end labels become final states (with added self transition)
- Possible runs of LTS correspond to accepted words over variables

SPIN Model Checking—Overview

$$\mathcal{T} \models \phi \quad ?$$

1. Represent transition system $\mathcal{T}$ as Büchi automaton $\mathcal{B}_\mathcal{T}$ such that $\mathcal{B}_\mathcal{T}$ accepts exactly those words corresponding to runs through $\mathcal{T}$
2. Construct Büchi automaton $\mathcal{B}_{\neg\phi}$ for negation of formula ϕ
3. If

   $$\mathcal{L}^\omega(\mathcal{B}_\mathcal{T}) \cap \mathcal{L}^\omega(\mathcal{B}_{\neg\phi}) = \emptyset$$

   then ϕ holds, otherwise we have a counterexample

   To check $\mathcal{L}^\omega(\mathcal{B}_\mathcal{T}) \cap \mathcal{L}^\omega(\mathcal{B}_{\neg\phi})$ construct intersection automaton and search for cycle through accepting state

Literature for this Lecture

Ben-Ari Section 5.2.1
(only syntax of LTL)

Baier and Katoen Principles of Model Checking, May 2008,
The MIT Press, ISBN: 0-262-02649-X
Vorhanden in ULB