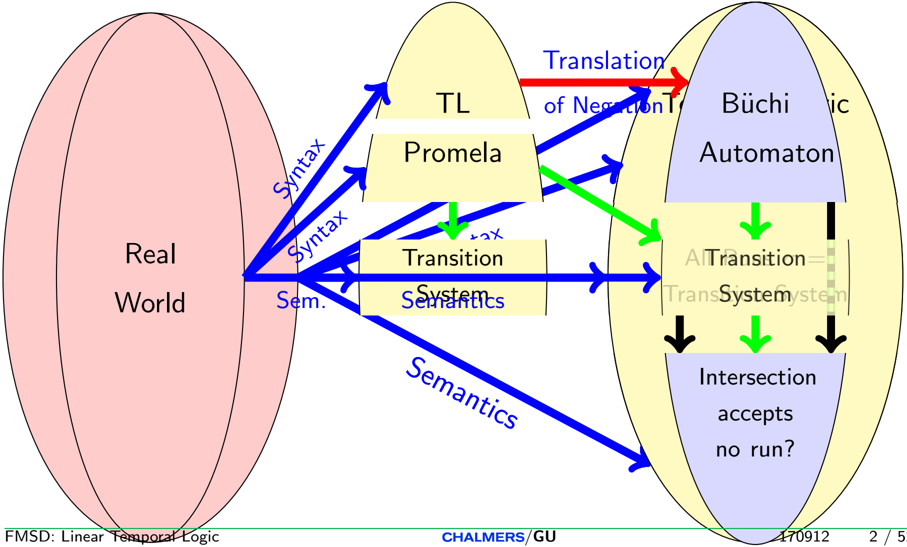
Formal Methods for Software Development

Propositional and (Linear) Temporal Logic

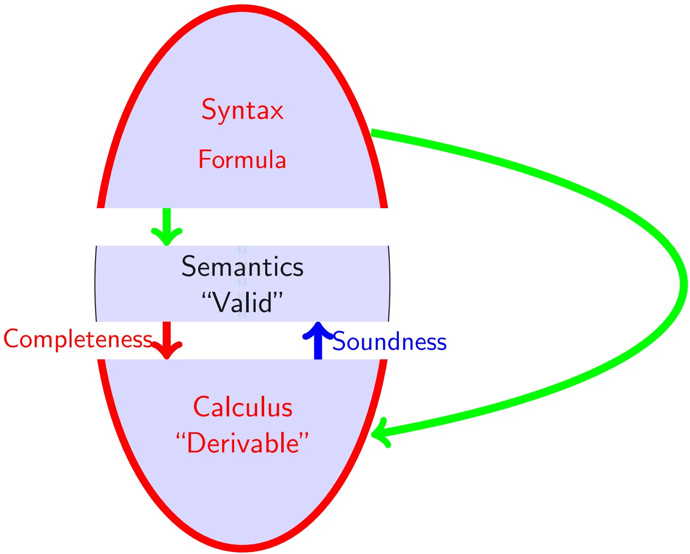
Wolfgang Ahrendt

12th September 2017

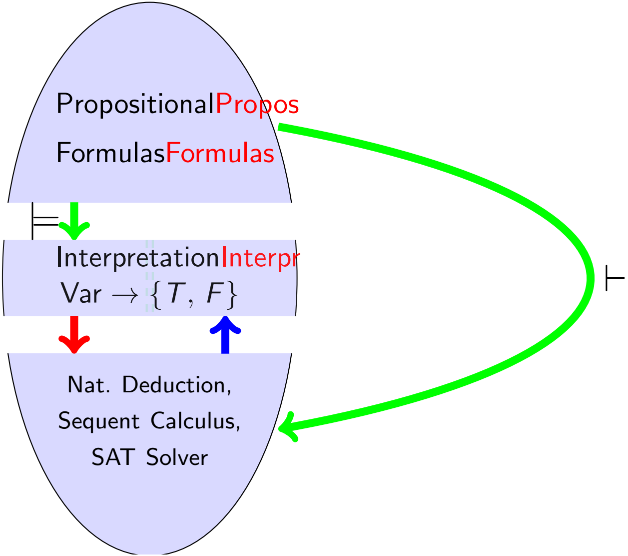
Recapitulation: FormalisationFormalisation: Syntax, SemanticsFormalisation: Syntax, Semantics, ProvingFormal Verification: Model Checking



The Big Picture: Syntax, Semantics, Calculus



Simplest Case: Propositional Logic—Syntax



Syntax of Propositional Logic

Signature

A set of Propositional Variables AP
('atomic propositions', with typical elements $p, q, r, \ldots$)

Propositional Connectives

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

Set of Propositional Formulas For_0

- Truth constants true, false and variables AP are formulas
- If ϕ and ψ are formulas then

  $$\neg\phi, \quad \phi \wedge \psi, \quad \phi \vee \psi, \quad \phi \rightarrow \psi, \quad \phi \leftrightarrow \psi$$

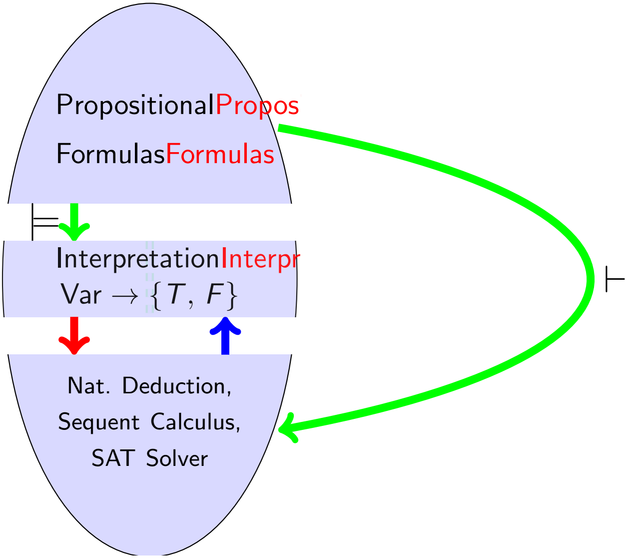
  are also formulas
- There are no other formulas (inductive definition)

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.

Simplest Case: Propositional Logic—Syntax



Semantics of Propositional Logic

Interpretation $\mathcal{I}$

Assigns a truth value to each propositional variable

$$\mathcal{I} : AP \rightarrow \{T, F\}$$

Example

Let $AP = \{p, q\}$

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

	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

$$\mathcal{I} : AP \rightarrow \{T, F\}$$

Valuation Function

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

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

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

(cont'd next page)

Semantics of Propositional Logic (Cont'd)

Valuation function (Cont'd)

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

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

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

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

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

Valuation Examples

Example

Let $AP = \{p, q\}$

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

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

$$\ldots$$

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

$$\begin{aligned}
val_{\mathcal{I}_2}(\, p \rightarrow (q \rightarrow p)\,) &= T \text{ iff } val_{\mathcal{I}_2}(p) = F \textbf{ or } val_{\mathcal{I}_2}(q \rightarrow p) = T \\
val_{\mathcal{I}_2}(p) &= \mathcal{I}_2(p) = T \\
val_{\mathcal{I}_2}(\, q \rightarrow p\,) &= T \text{ iff } val_{\mathcal{I}_2}(q) = F \textbf{ or } val_{\mathcal{I}_2}(p) = T \\
val_{\mathcal{I}_2}(q) &= \mathcal{I}_2(q) = F
\end{aligned}$$

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 $\mathcal{I}$:

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

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?

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

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, not valid!

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

Does it hold? Yes. Why?

An Exercise in Formalisation

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

Can we characterise the states of P propositionally?

Find a propositional formula ϕ_{P} which is true if and only if it describes a possible state of P.

$$\phi_{\mathrm{P}} := \left(\begin{array}{l} ((PC0_3 \wedge \neg PC0_4 \wedge \neg PC0_5) \vee \ldots) \wedge \\ ((\neg PC0_5 \wedge \neg PC1_5) \implies (\neg N_0 \wedge \ldots \wedge \neg N_7)) \wedge \ldots \end{array} \right)$$

An Exercise in Formalisation

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

AP : $N_0, N_1, N_2, \ldots, N_7$ 8-bit representation of **byte**
$PC0_3, PC0_4, PC0_5, PC1_3, PC1_4, PC1_5$ next instruction pointer

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
- ...

$$\phi_{\mathrm{P}} := \left(\begin{array}{l} ((PC0_3 \land \neg PC0_4 \land \neg PC0_5) \lor \ldots) \land \\ ((\neg PC0_5 \land \neg PC1_5) \implies (\neg N_0 \land \ldots \land \neg N_7)) \land \ldots \end{array} \right)$$

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 any possible state reachable in any run of P

But How to Express Properties Involving State Changes?

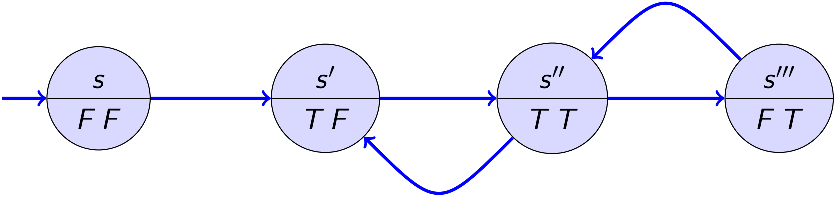
In any run of a program P

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

etc.

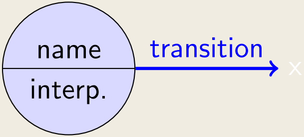
$\Rightarrow$ Need a more expressive logic: (Linear) Temporal Logic

Transition Systems (aka Kripke Structures)

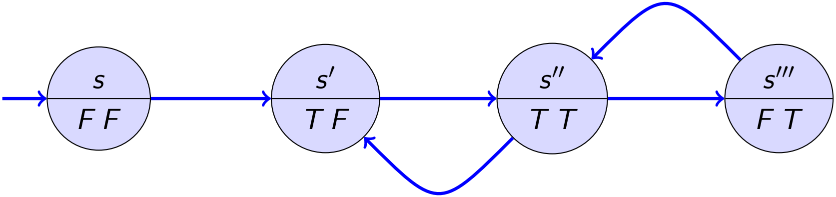


We assume $AP = \{p, q\}$

Notation

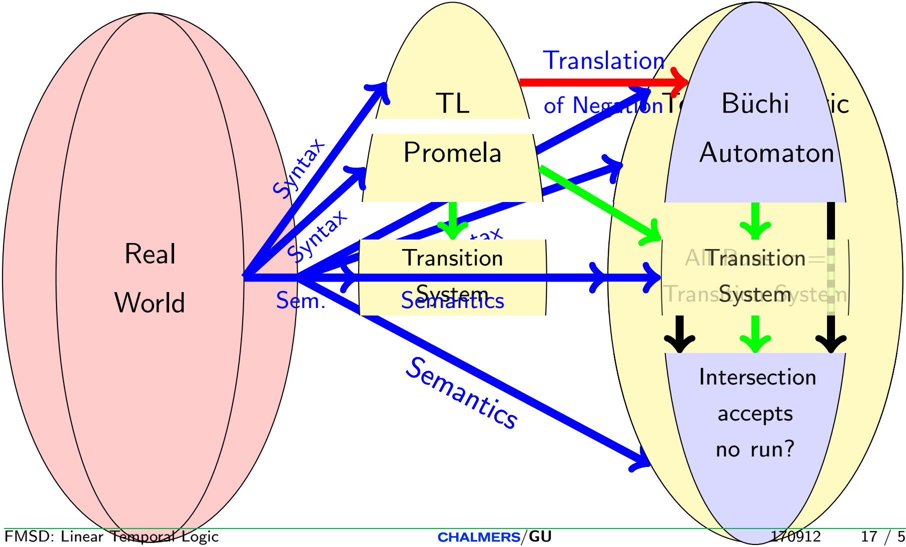


Transition Systems (aka Kripke Structures)

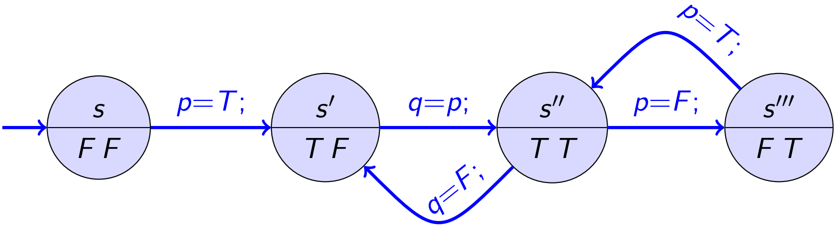


- Each state has *its own* interpretation $\mathcal{I} : \{p, q\} \to \{T, F\}$
  - Convention: list interpretation of variables in lexicographic order
- Computations, or runs, are *infinite* paths through states
  - 'finite' runs simulated by looping on terminal state
- Prefix of some example runs:
  - $s\, s' s'' s' s'' s' s'' s''' \ldots$
  - $s\, s' s'' s''' s'' s' s'' s' \ldots$

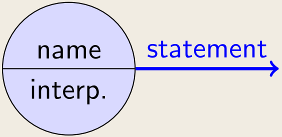
Recapitulation: FormalisationFormalisation: Syntax, SemanticsFormalisation: Syntax, Semantics, ProvingFormal Verification: Model Checking



Transition System of some PROMELA Model



Notation



Transition Systems: Formal Definition

Definition (Transition System)

A transition system $\mathcal{T} = (S, \rightarrow, S_o, L)$ is composed of a set of states S, a transition relation $\rightarrow \subseteq S \times S$, a set $\emptyset \neq S_0 \subseteq S$ of initial states, and a labeling L of each state $s \in S$ with a propositional interpretation $L(s)$.

Definition (Run of Transition System)

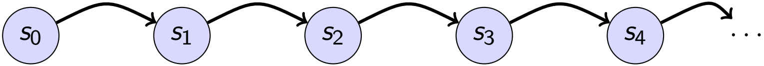
A run of $\mathcal{T} = (S, \rightarrow, S_o, L)$ is a sequence of states
$\sigma = s_0\, s_1 \ldots$
such that $s_0 \in S_0$ and $s_i \rightarrow s_{i+1}$ for all $i \geq 0$.

Definition (Trace)

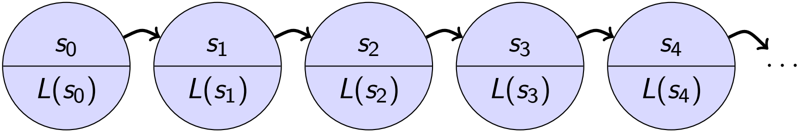
The trace $tr(\sigma)$ of a run $\sigma = s_0\, s_1 \ldots$ is the sequence
$\tau = \mathcal{I}_0\, \mathcal{I}_1 \ldots$
such that $\mathcal{I}_i = L(s_i)$.
A trace of $\mathcal{T}$ is $tr(\sigma)$ for any run σ of $\mathcal{T}$.

Runs and Traces Visually

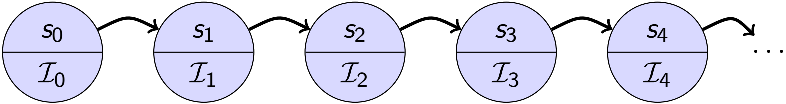
- Given a run $\sigma = s_0\, s_1\, s_2\, s_3\, s_4 \ldots$



- Each state s of a transition system is labelled, via $L(s)$, with an interpretation



- If we name each interpretations $L(s_i)$ as $\mathcal{I}_i$, we have



- The trace $tr(\sigma)$ is: $\tau = \mathcal{I}_0\, \mathcal{I}_1\, \mathcal{I}_2\, \mathcal{I}_3 \ldots$

Notations: Power Set and Sequences

Assume sets X and Y.

Power Set

2^X is the set of all subsets of X (called 'power set of X').

Finite Sequences

Y^* is the set of all finite sequences (words) of elements of Y.

Infinite Sequences

Y^ω is the set of all infinite sequences (words) of elements of Y.

Power Sets and Sequences: Example

Given the set of atomic propositions $AP = \{p, q\}$.

Power Set

$2^{AP} = \{\,\{\}, \{p\}, \{p\}, \{p, q\}\,\}$

Finite Sequences

$(2^{AP})^*$: set of all finite sequences of elements of 2^{AP}.
E.g.: $\{p\}\{\}\{p, q\}\{p\} \in (2^{AP})^*$

(and infitely many others)

Infinite Sequences

$(2^{AP})^\omega$: set of all infinite sequences of elements of 2^{AP}.
E.g.: $\{p\}\{p, q\}\{p\}\{\}\{p\}\{p, q\}\{p\}\{\} \ldots \in (2^{AP})^\omega$

(and uncountably many others)

Interpretations as Sets

Interpretations over atomic propositions AP can be represented as elements of 2^{AP}.

E.g., assume $AP = \{p, q\}$
I.e., $2^{AP} = \{\,\{\}, \{p\}, \{p\}, \{p, q\}\,\}$

	p	q
$\mathcal{I}_1$	F	F

represented as $\{\}$

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

represented as $\{p\}$

	p	q
$\mathcal{I}_3$	F	T

represented as $\{q\}$

	p	q
$\mathcal{I}_4$	T	T

represented as $\{p, q\}$

Runs and Traces revisited

Given states S and atomic propositions AP.

- A run $\sigma = s_0\, s_1\, s_2\, s_3\, s_4 \ldots$ is an element of S^ω.
- A trace $\tau = \mathcal{I}_0\, \mathcal{I}_1\, \mathcal{I}_2\, \mathcal{I}_3 \ldots$ is an element of of $(2^{AP})^\omega$.

An example of a trace $\tau = \mathcal{I}_0\, \mathcal{I}_1\, \mathcal{I}_2\, \mathcal{I}_3 \ldots$ may look like:

$\tau = \{p\}\{p, q\}\{p\}\{\} \ldots$

Linear Time Properties

Definition (Linear Time Property)

Given a set of atomic propositions AP.
Each subset P of $(2^{AP})^{\omega}$ is a linear time (LT) property over AP.

Intuition:

- Assume a trace property $P \subseteq (2^{AP})^{\omega}$.
- A trace t fulfils the property P iff $t \in P$.
- A trace t violates the property P iff $t \notin P$.

Classes of LT Properties

The LT properties can be devided in three classes:

- Safety properties
- Liveness properties
- Properties that are neither safety nor liveness properties

Safety Properties

Definition (Safety Properties, Bad Prefixes)

An LT property P_{safe} over AP is called a safety property if for all words $\tau \in (2^{AP})^\omega \setminus P_{safe}$, there exists a finite prefix $\hat{\tau}$ of τ such that

$$P_{safe} \cap \left\{ \tau' \in (2^{AP})^\omega \mid \hat{\tau} \text{ is a finite prefix of } \tau' \right\} = \emptyset$$

Each violating trace τ has a finite, 'bad prefix' $\hat{\tau}$.

Liveness Properties

Let $pref(P)$ be the set of finite prefixes of elements of P.

Definition (Liveness Properties)

An LT property P_{live} over AP is called a liveness property whenever $pref(P_{live}) = (2^{AP})^*$

A liveness property allows every finite prefix.
(It cannot be refuted in finite time.)

Linear Temporal Logic—Syntax

An extension of propositional logic that
allows to specify properties of all traces

Syntax

Based on propositional signature and syntax

Extension with three connectives (in this course):

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

Eventually 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	$\Box$	`[]`
Eventually	$\Diamond$	`<>`
Until	$\mathcal{U}$	`U`

Linear Temporal Logic Syntax: Examples

Let $AP = \{p, q\}$ be the set of propositional variables.

- p
- false
- $p \to q$
- $\Diamond p$
- $\Box q$
- $\Diamond \Box (p \to q)$
- $(\Box p) \to ((\Diamond p) \vee \neg q)$
- $p \mathcal{U} (\Box q)$

Temporal Logic—Semantics

Valuation of temporal formula relative to trace (infinite sequence of interpretations)

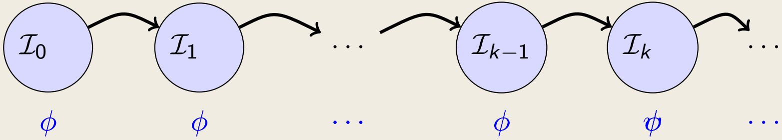
Definition (Validity Relation)

Validity of temporal formula depends on traces $\tau = \mathcal{I}_0 \, \mathcal{I}_1 \ldots$

$\tau \models p$	iff	$\mathcal{I}_0(p) = T$, for $p \in AP$.
$\tau \models \neg\phi$	iff	not $\tau \models \phi$ (write $\tau \not\models \phi$)
$\tau \models \phi \wedge \psi$	iff	$\tau \models \phi$ and $\tau \models \psi$
$\tau \models \phi \vee \psi$	iff	$\tau \models \phi$ or $\tau \models \psi$
$\tau \models \phi \rightarrow \psi$	iff	$\tau \not\models \phi$ or $\tau \models \psi$

Temporal connectives?

Trace τ



If $\tau = \mathcal{I}_0\,\mathcal{I}_1 \ldots$, then $\tau|_i$ denotes the suffix $\mathcal{I}_i\,\mathcal{I}_{i+1} \ldots$ of τ.

Definition (Validity Relation for Temporal Connectives)

Given a trace $\tau = \mathcal{I}_0\,\mathcal{I}_1 \ldots$

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

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

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

Safety and Liveness Properties

Safety Properties

- Always-formulas called safety properties:
  "something bad never happens"
- Example:
  $\Box(\neg\texttt{P\_in\_CS} \lor \neg\texttt{Q\_in\_CS})$
  'simultaneous visit to the critical sections never happens'

Liveness Properties

- Eventually-formulas called liveness properties:
  "something good happens eventually"
- Example:
  $\Diamond\,\texttt{P\_in\_CS}$
  'P enters its critical section eventually'

Complex Properties

What does this mean?Infinitely Often

$$\tau \models \Box\Diamond\phi$$

"During trace τ the formula ϕ becomes true infinitely often"

Validity of Temporal Logic

Definition (Validity)

ϕ is valid, write $\models \phi$, iff $\tau \models \phi$ for all traces $\tau = \mathcal{I}_0\,\mathcal{I}_1 \ldots$.

Representation of Traces

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

- $\phi_0\,\phi_1, \ldots$ represents all traces $\mathcal{I}_0\,\mathcal{I}_1 \ldots$ such that $\mathcal{I}_i \models \phi_i$ for $i \geq 0$

Semantics of Temporal Logic: Examples

$$\Diamond\Box\phi$$

Valid?

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

Valid in some trace?

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

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

All are valid! (proof is exercise)

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

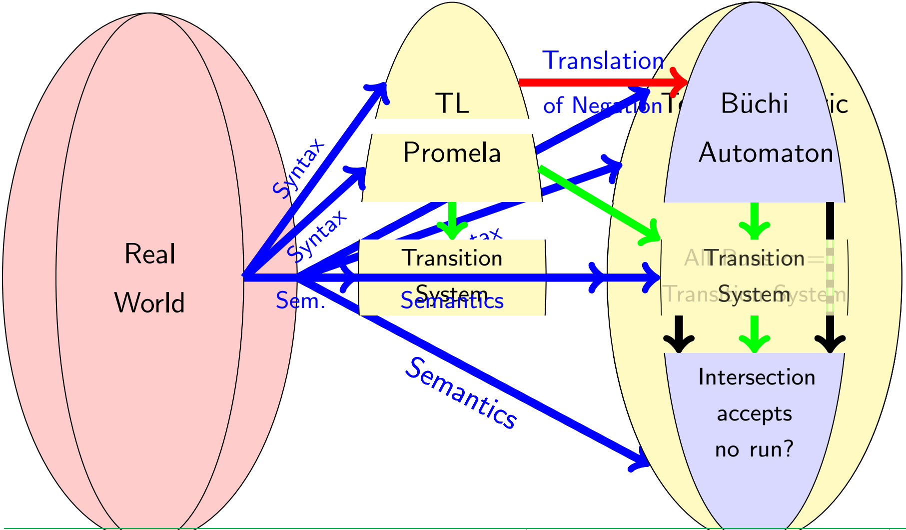
Temporal Logic—Semantics (Cont'd)

Extension of validity of temporal formulas to transition systems:

Definition (Validity Relation)

Given a transition system $\mathcal{T} = (S, \rightarrow, S_0, L)$, a temporal formula ϕ is valid in $\mathcal{T}$ (write $\mathcal{T} \models \phi$) iff $\tau \models \phi$ for all traces τ of $\mathcal{T}$.

Recapitulation: FormalisationFormalisation: Syntax, SemanticsFormalisation: Syntax, Semantics, ProvingFormal Verification: Model Checking



ω-Languages

Given a finite alphabet (vocabulary) Σ

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

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

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

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

Büchi Automaton

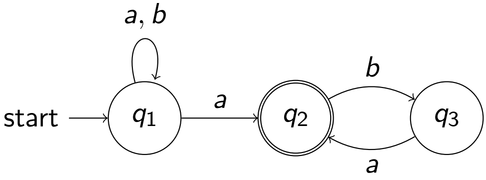
Definition (Büchi Automaton)

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

- finite, non-empty set of locations Q
- a transition relation $\delta \subseteq Q \times \Sigma \times Q$
- a non-empty set of initial locations $Q_0 \subseteq Q$
- a set of accepting locations $F = \{f_1, \ldots, f_n\} \subseteq Q$

Example

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



Büchi Automaton—Executions and Accepted Words

Definition (Execution)

Let $\mathcal{B} = (Q, \delta, Q_0, F)$ be a Büchi automaton over alphabet Σ.
An execution of $\mathcal{B}$ is a pair (w, v), with

- $w = a_o \ldots a_k \ldots \in \Sigma^\omega$
- $v = q_o \ldots q_k \ldots \in Q^\omega$

where $q_0 \in Q_0$, and $(q_i, a_i, q_{i+1}) \in \delta$, for all $i \in \mathbb{N}$

Definition (Accepted Word)

A Büchi automaton $\mathcal{B}$ accepts a word $w \in \Sigma^\omega$, if there exists an execution (w, v) of $\mathcal{B}$ where some accepting location $f \in F$ appears infinitely often in v.

Büchi Automaton—Language

Let $\mathcal{B} = (Q, \delta, Q_0, F)$ be a Büchi automaton, then

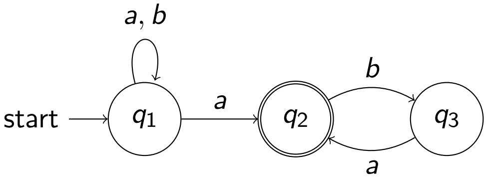
$$\mathcal{L}^\omega(\mathcal{B}) = \{ w \in \Sigma^\omega \,|\, \mathcal{B} \text{ accepts } w \}$$

denotes the ω-language recognised 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 similar to standard regular expression

ab a **followed by** 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:

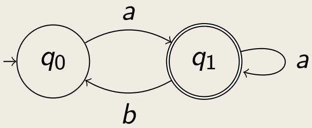
- *if $\mathcal{L}_1, \mathcal{L}_2$ are ω-regular then $\mathcal{L}_1 \cap \mathcal{L}_2$ and $\mathcal{L}_1 \cup \mathcal{L}_2$ are ω-regular*
- *$\mathcal{L}$ is ω-regular then $\Sigma^\omega \backslash \mathcal{L}$ is ω-regular*

But in contrast to regular finite automata:

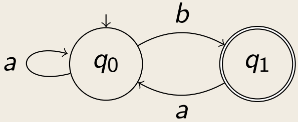
Non-deterministic Büchi automata are strictly more expressive than deterministic ones.

Büchi Automata—More Examples

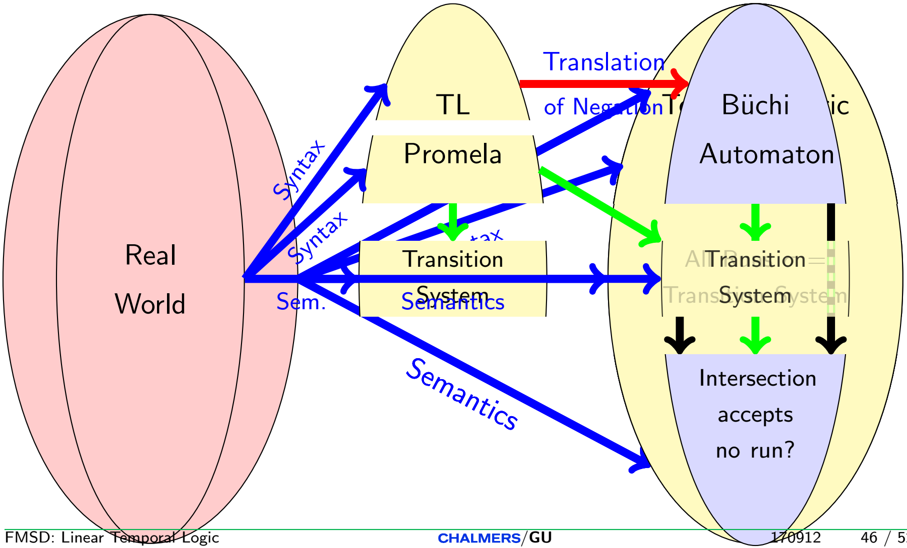
Language: $a(a + ba)^\omega$



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



Recapitulation: FormalisationFormalisation: Syntax, SemanticsFormalisation: Syntax, Semantics, ProvingFormal Verification: Model Checking



Linear Temporal Logic and Büchi Automata

LTL and Büchi Automata are connected

Recall

Definition (Validity Relation)

Given a transition system $\mathcal{T} = (S, \rightarrow, S_0, L)$, a temporal formula ϕ is valid in $\mathcal{T}$ (write $\mathcal{T} \models \phi$) iff $\tau \models \phi$ for all traces τ of $\mathcal{T}$.

A trace of the transition system is an infinite sequence of interpretations.

Intended Connection

Given an LTL formula ϕ:

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

Encoding an LTL Formula as a Büchi Automaton

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

Suitable alphabet Σ for Büchi automaton?

A state transition of Büchi automaton must represent an interpretation

Choose Σ to be the set of all interpretations over AP, encoded as 2^{AP}.

(Recall slide 'Interpretations as Sets')

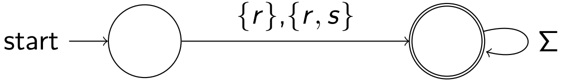
Example

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

Büchi Automaton for LTL Formula By Example

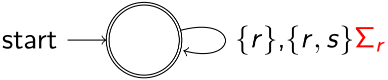
Example (Büchi automaton for formula r over $AP = \{r, s\}$)

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



In the first state s_0 (of σ) at least r must hold, the rest is arbitrary

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

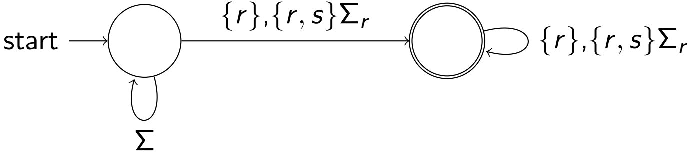


$$\Sigma_r := \{I | I \in \Sigma, r \in I\}$$

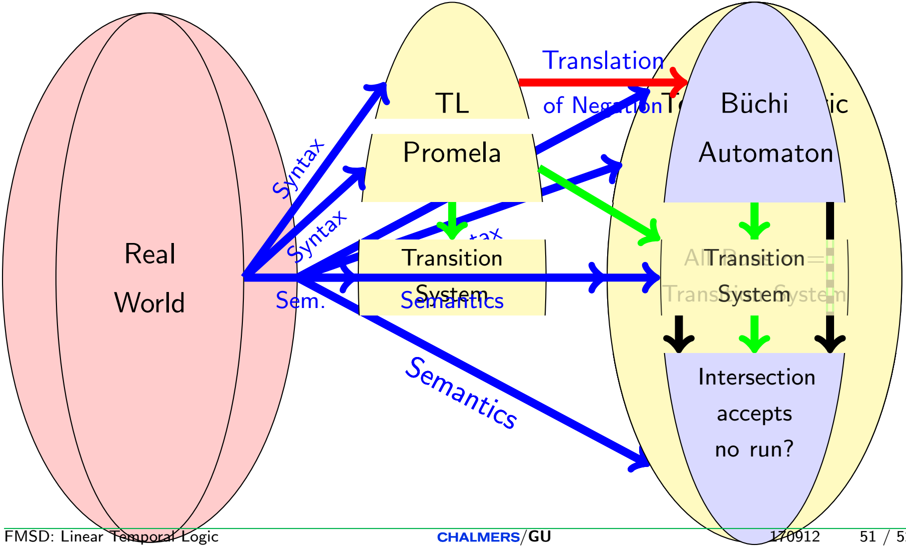
In *all* states s (of σ) at least r must hold

Büchi Automaton for LTL Formula By Example

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



Recapitulation: FormalisationFormalisation: Syntax, SemanticsFormalisation: Syntax, Semantics, ProvingFormal Verification: Model Checking



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
(for in depth theory of model checking)