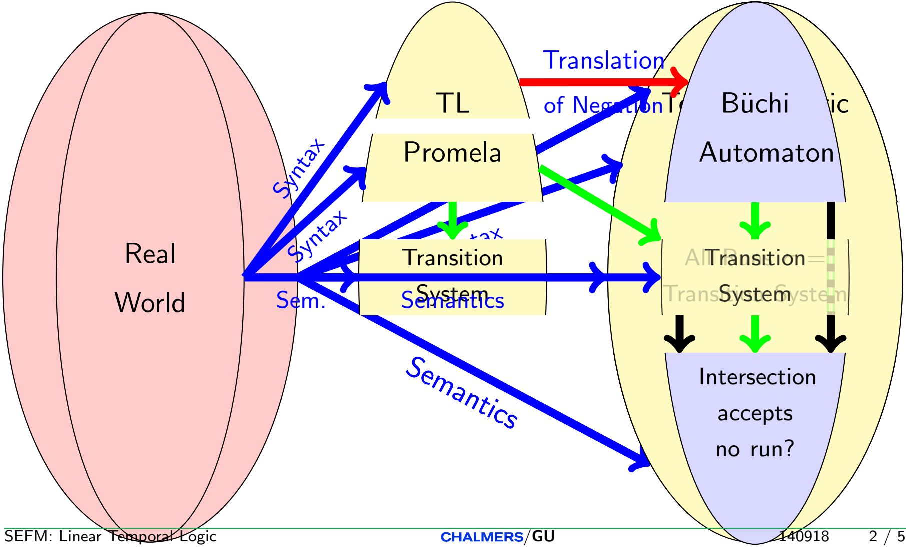
Software Engineering using Formal Methods

Formal Modeling with Linear Temporal Logic

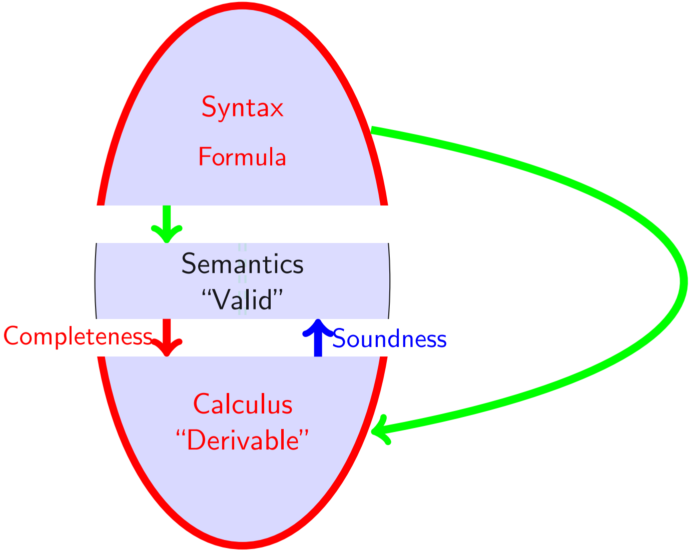
Wolfgang Ahrendt

18th September 2014

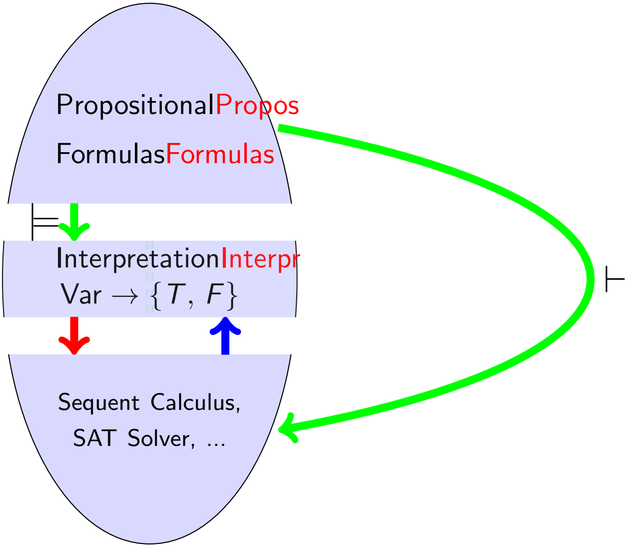
Recapitulation: FormalisationFormalisation: Syntax, SemanticsFormalisation: Syntax, Semantics, ProvingFormal Verification: Model Checking



The Big Picture: Syntax, Semantics, Calculus



Simplest Case: Propositional Logic—Syntax



Syntax of Propositional Logic

Signature

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

Propositional Connectives

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

Set of Propositional Formulas For_0

- Truth constants true, false and variables $\mathcal{P}$ 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

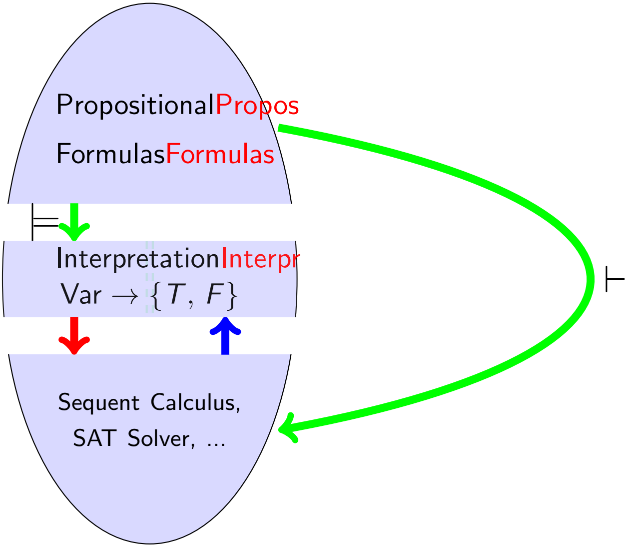
Propositional Logic Syntax: Examples

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

Are the following character sequences also propositional formulas?

- $\mathrm{true} \rightarrow p$ ✔
- $(p(q \land r)) \lor p$ ✘
- $p \rightarrow (q\land)$ ✘
- $\mathit{false} \land (p \rightarrow (q \land r))$ ✔

Simplest Case: Propositional Logic—Syntax



Semantics of Propositional Logic

Interpretation $\mathcal{I}$

Assigns a truth value to each propositional variable

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

Example

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

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

	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

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

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

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

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

$\ldots$

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

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

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, also 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 (iff) it describes a possible state of P.

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

An Exercise in Formalisation

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

$\mathcal{P}$: $N_0, N_1, N_2, \ldots, N_7$ 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
- ...

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

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?

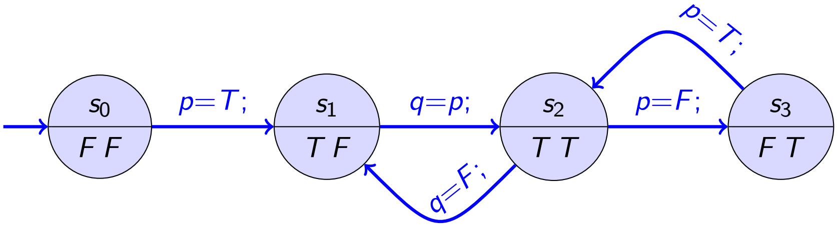
In any run of a program P

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

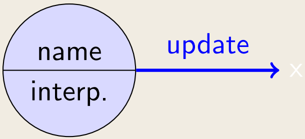
etc.

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

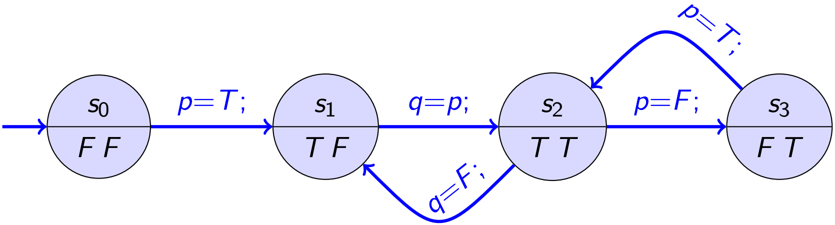
Transition systems (aka Kripke Structures)



Notation

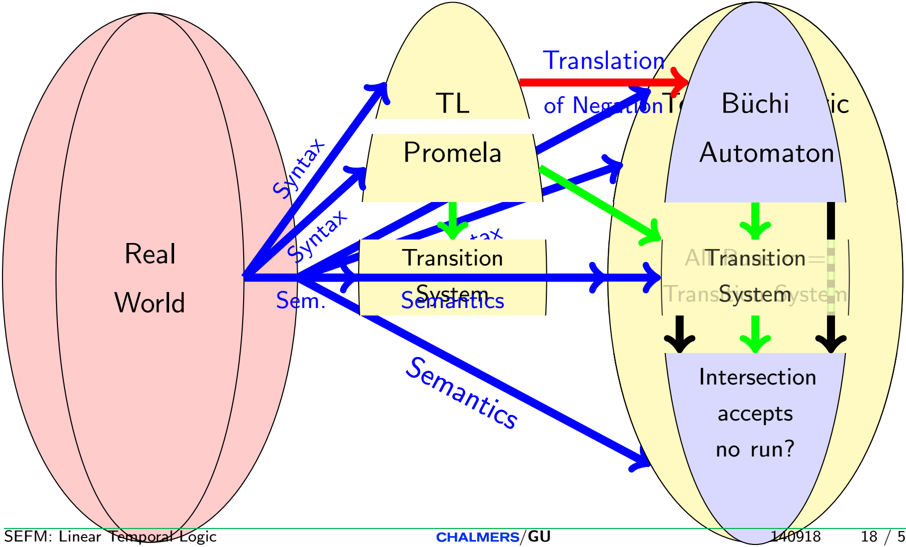


Transition systems (aka Kripke Structures)



- Each state s_i has its own propositional interpretation I_i
  - Convention: list values of variables in ascending lexicographic order
- Computations, or runs, are *infinite* paths through states
  - Intuitively 'finite' runs modelled by looping on last state
- How to express (for example) that p changes its value infinitely often in each run?

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



(Linear) Temporal Logic—Syntax

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

Syntax

Based on propositional signature and syntax

Extension with three connectives:

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

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

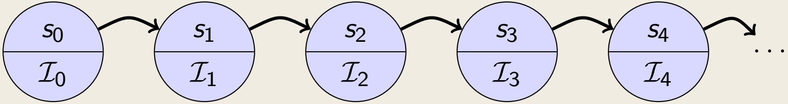
Until If ϕ and ψ are formulas then so is $\phi\mathcal{U}\psi$

Concrete Syntax

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

Temporal Logic—Semantics

A run σ is an infinite chain of states



$\mathcal{I}_j$ propositional interpretation of variables in j-th state
Write 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)

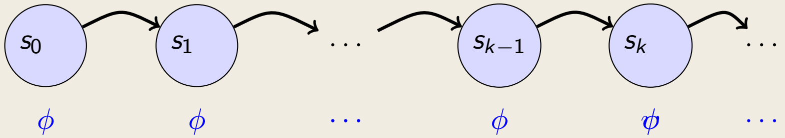
Validity of temporal formula depends on runs $\sigma = s_0\, s_1 \ldots$

$\sigma \models p$	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 $\sigma \models \psi$
$\sigma \models \phi \rightarrow \psi$	iff	$\sigma \not\models \phi$ or $\sigma \models \psi$

Temporal connectives?

Temporal Logic—Semantics (Cont'd)

Run σ



Definition (Validity Relation for Temporal Connectives)

Given a run $\sigma = s_0 \, s_1 \cdots$

$\sigma \models \Box\phi$ 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 properties:
  "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
- $\Box$ `mutex` expresses that simultaneous access never happens

Liveness Properties

- Eventually-formulas called liveness properties:
  "something good happens eventually"
- Let `s` be variable that is true when a process delivers a service
- $\Diamond$ `s` expresses that service is eventually provided

Complex Properties

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 $\sigma \models \phi$ for 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 $s_i \models \phi_i$ for $i \geq 0$

Semantics of Temporal Logic: Examples

$$\Diamond\Box\phi$$

Valid?

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

Valid in some run?

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 $\mathcal{U}$

Transition Systems: Formal Definition

Definition (Transition System)

A transition system $\mathcal{T} = (S, Ini, \delta, \mathcal{I})$ is composed of a set of states S, a set $\emptyset \neq Ini \subseteq S$ of initial states, a transition relation $\delta \subseteq S \times S$, and a 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 $s_i \in S$ as well as $(s_i, s_{i+1}) \in \delta$.

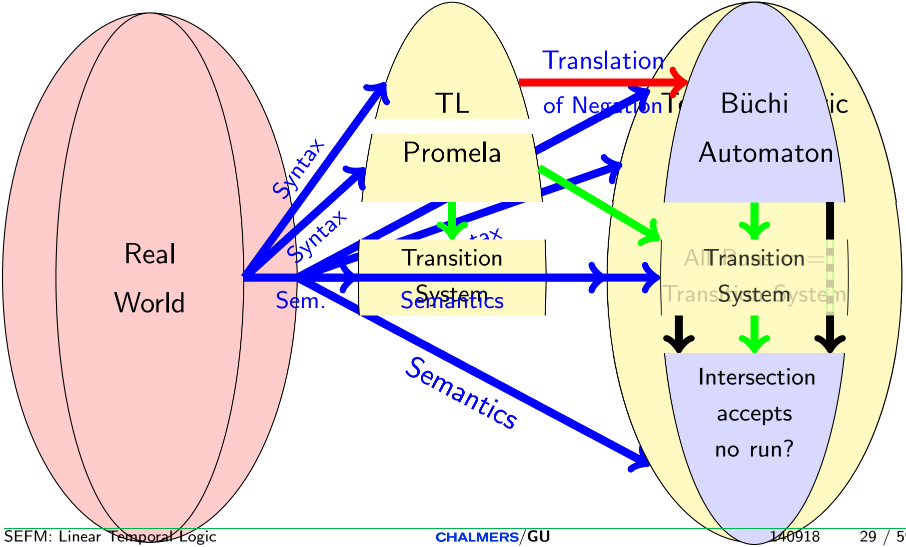
Temporal Logic—Semantics (Cont'd)

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 $\mathcal{T}$.

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

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

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

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

Büchi Automaton

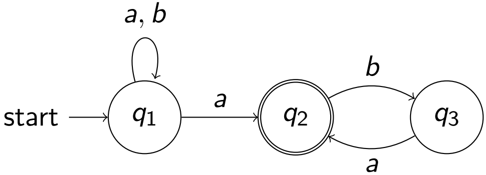
Definition (Büchi Automaton)

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

- finite, non-empty set of locations Q
- a non-empty set of initial/start locations $I \subseteq Q$
- a set of accepting 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—Executions and Accepted Words

Definition (Execution)

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

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

where $q_0 \in I$, 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 $\mathcal{B} = (Q, I, F, \delta)$ be a Büchi automaton, then

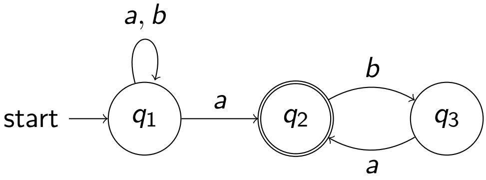
$$\mathcal{L}^\omega(\mathcal{B}) = \{w \in \Sigma^\omega | w \in \Sigma^\omega \text{ is an accepted word of } \mathcal{B}\}$$

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 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:

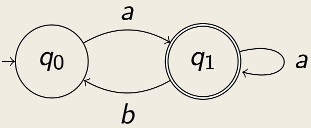
- *if $\mathcal{L}_1, \mathcal{L}_2$ are ω-regular then $\mathcal{L}_1 \cap \mathcal{L}_2$ and $\mathcal{L}_1 \cup \mathcal{L}_2$ are ω-regular*
- *$\mathcal{L}$ is ω-regular then $\Sigma^\omega \setminus \mathcal{L}$ is ω-regular*

But in contrast to regular finite automata

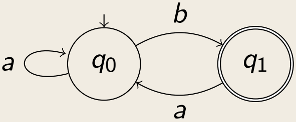
Non-deterministic Büchi automata are strictly more expressive than deterministic ones

Büchi Automata—More Examples

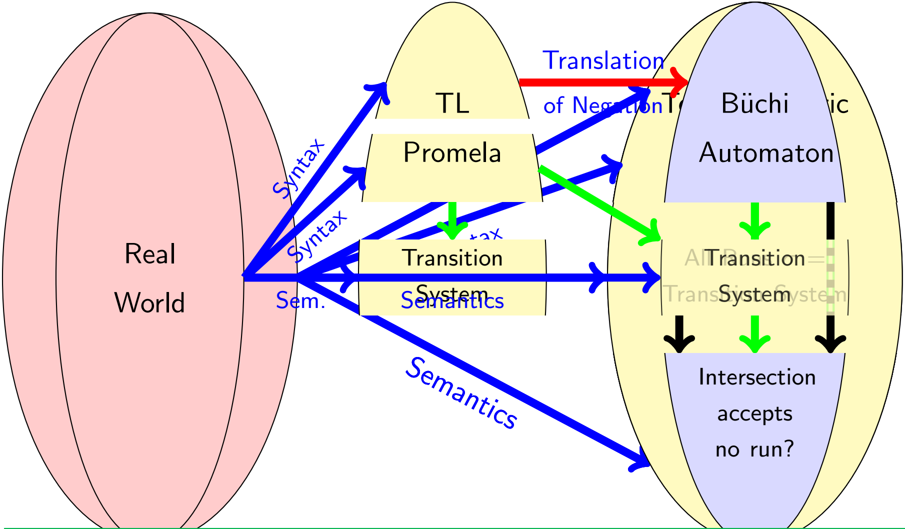
Language: $a(a + ba)^\omega$



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 $\mathcal{T} = (S, Ini, \delta, \mathcal{I})$, 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., $\mathcal{P} = \{r, s\}$

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 $\mathcal{P}$, encoded as $2^{\mathcal{P}}$

Example

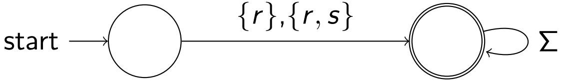
$\Sigma = \big\{\emptyset, \{r\}, \{s\}, \{r, s\}\big\}$

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

Büchi Automaton for LTL Formula By Example

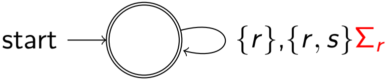
Example (Büchi automaton for formula r over $\mathcal{P} = \{r, s\}$)

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



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

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

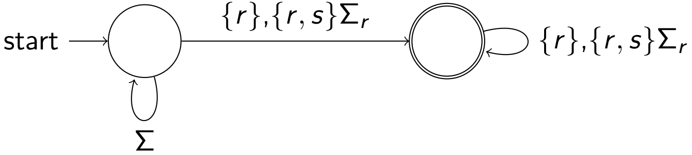


$$\Sigma_r := \{I | I \in \Sigma, r \in I\}$$

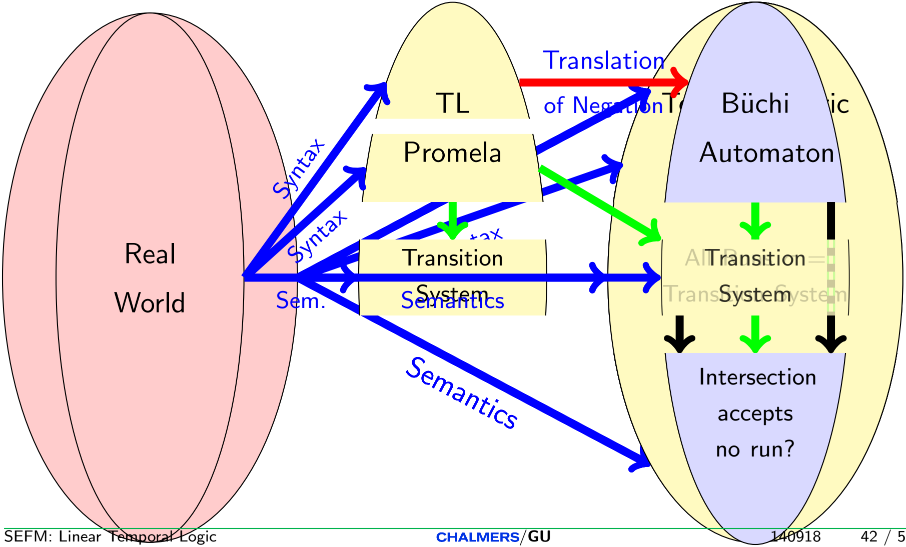
In *all* states s (of σ) at least r must hold

Büchi Automaton for LTL Formula By Example

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



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



Model Checking

Check whether a 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

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 $\mathcal{T} \models \phi$ holds.

   If

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

   then each element of the set is a counterexample for ϕ.

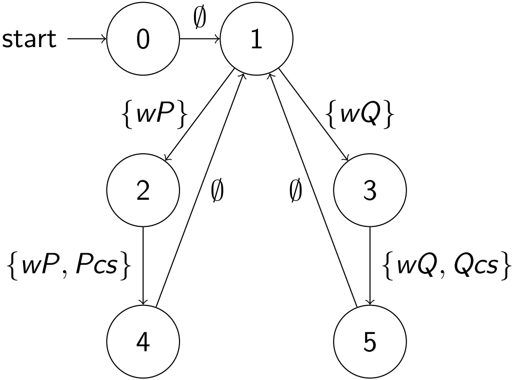
   To check $\mathcal{L}^{\omega}(\mathcal{B}_{\mathcal{T}}) \cap \mathcal{L}^{\omega}(\mathcal{B}_{\neg\phi})$ construct intersection automaton and search for cycle through accepting state

Representing a Model as a Büchi Automaton

First Step: Represent transition system $\mathcal{T}$ as Büchi automaton $\mathcal{B}_{\mathcal{T}}$ accepting exactly those words representing a run of $\mathcal{T}$

Example

```
active proctype P () {
do
  :: atomic {
      !wQ; wP = true
     };
     Pcs = true;
     atomic {
      Pcs = false;
      wP = false
     }
od }
```



Similar code for process `Q`.

Second `atomic` block just to keep automaton small.

Büchi Automaton $B_{\neg\phi}$ for $\neg\phi$

Second Step:

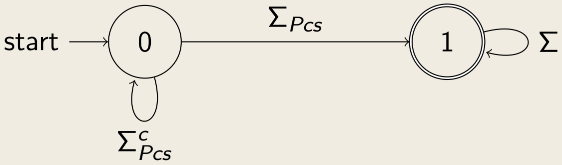
Construct Büchi Automaton corresponding to negated LTL formula

$\mathcal{T} \models \phi$ holds iff there is no accepting run σ of $\mathcal{T}$ s.t. $\sigma \models \neg\phi$

Simplify $\neg\phi = \neg\Box\neg Pcs = \Diamond Pcs$

Büchi Automaton $\mathcal{B}_{\neg\phi}$

$$\mathcal{P} = \{wP, wQ, Pcs, Qcs\},\ \Sigma = 2^{\mathcal{P}}$$



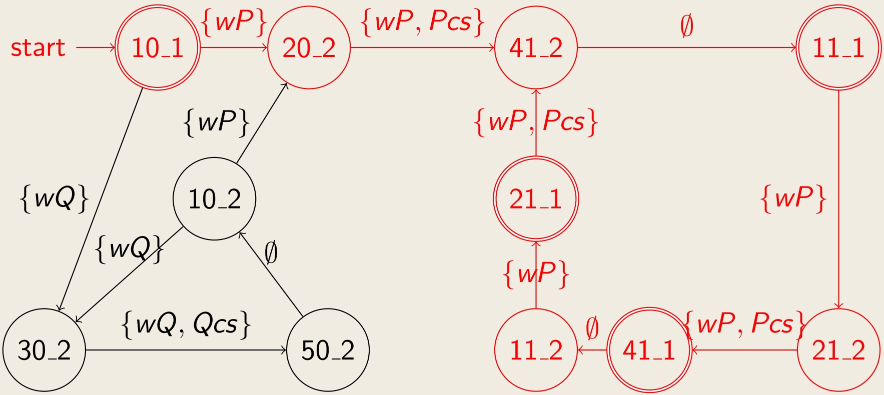
$$\Sigma_{Pcs} = \{I | I \in \Sigma, Pcs \in I\}, \quad \Sigma^c_{Pcs} = \Sigma - \Sigma_{Pcs}$$

Checking for Emptiness of Intersection Automaton

Third Step: $\mathcal{L}^\omega(\mathcal{B}_\mathcal{T}) \cap \mathcal{L}^\omega(\mathcal{B}_{\neg\phi}) = \neq \emptyset$?

Counterexample Construction of intersection automaton: Appendix

Intersection Automaton (skipping first step of $\mathcal{T}$ for simplicity)



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

Appendix I:

Intersection Automaton

–

Construction

Construction of Intersection Automaton

Given: two Büchi automata $\mathcal{B}_i = (Q_i, \delta_i, I_i, F_i)$, $i = 1, 2$

Wanted: a Büchi automaton

$$\mathcal{B}_{1\cap 2} = (Q_{1\cap 2}, \delta_{1\cap 2}, I_{1\cap 2}, F_{1\cap 2})$$

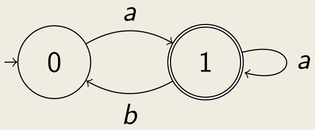
accepting a word w iff w is accepted by $\mathcal{B}_1$ and $\mathcal{B}_2$

Maybe just the product automaton as for regular automata?

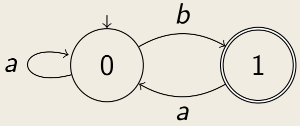
First Attempt: Product Automata for Intersection

$\Sigma = \{a, b\}$, $a(a + ba)^\omega \cap (a^* ba)^\omega = \emptyset$? No, e.g., $a(ba)^\omega$

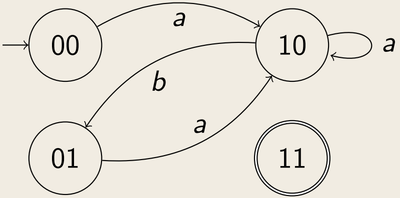
$a(a + ba)^\omega$:



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

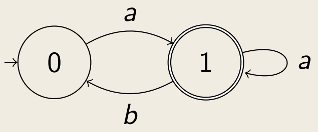


Product Automaton: accepting location 11 never reached

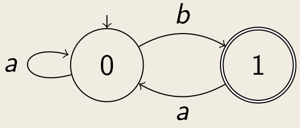


Explicit Construction of Intersection Automaton

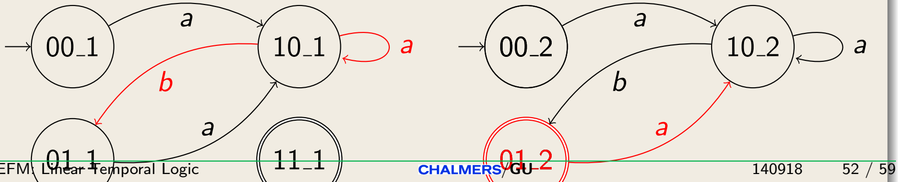
$a(a+ba)^\omega$:



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

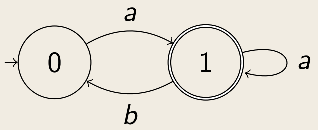


(i) Product Automaton(ii) Reachable States(iii) Clone(iv) Initial States Restricted to First Copy(v) Final States Restricted to First Atomaton of First Copy(vi) Ensure Acceptance in Both Copies $1 \to 2$**(vii) Ensure Acceptance in Both Copies** $2 \to 1$**(viii) Transitions of Product Automaton**

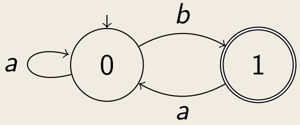


Explicit Construction of Intersection Automaton

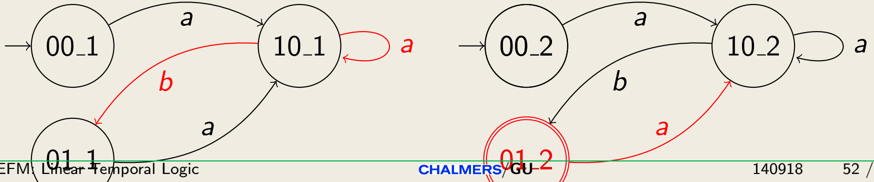
$a(a+ba)^\omega$:



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

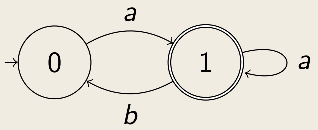


(i) Product Automaton(ii) Reachable States(iii) Clone(iv) Initial States Restricted to First Copy(v) Final States Restricted to First Atomaton of First Copy(vi) Ensure Acceptance in Both Copies $1 \rightarrow 2$**(vii) Ensure Acceptance in Both Copies** $2 \rightarrow 1$**(viii) Transitions of Product Automaton**

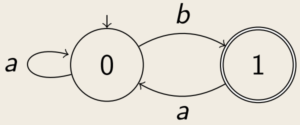


Explicit Construction of Intersection Automaton

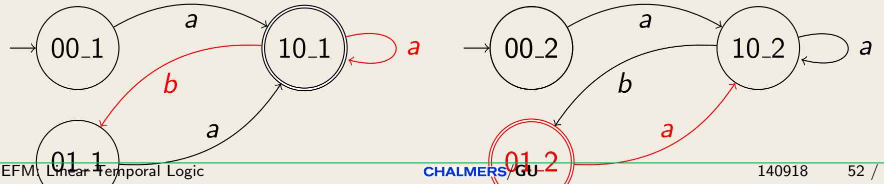
$a(a+ba)^\omega$:



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

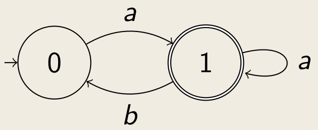


(i) Product Automaton(ii) Reachable States(iii) Clone(iv) Initial States Restricted to First Copy(v) Final States Restricted to First Atomaton of First Copy(vi) Ensure Acceptance in Both Copies $1 \to 2$**(vii) Ensure Acceptance in Both Copies** $2 \to 1$**(viii) Transitions of Product Automaton**

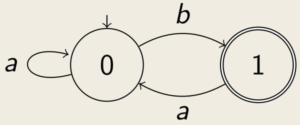


Explicit Construction of Intersection Automaton

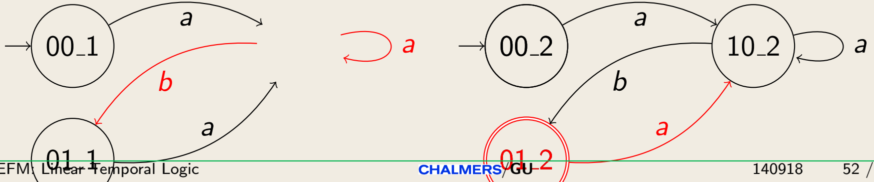
$a(a+ba)^\omega$:



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

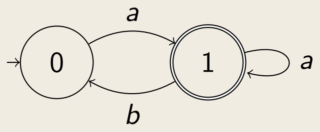


(i) Product Automaton(ii) Reachable States(iii) Clone(iv) Initial States Restricted to First Copy(v) Final States Restricted to First Atomaton of First Copy(vi) Ensure Acceptance in Both Copies $1 \rightarrow 2$**(vii) Ensure Acceptance in Both Copies** $2 \rightarrow 1$**(viii) Transitions of Product Automaton**

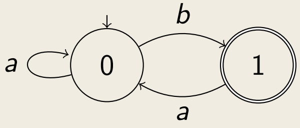


Explicit Construction of Intersection Automaton

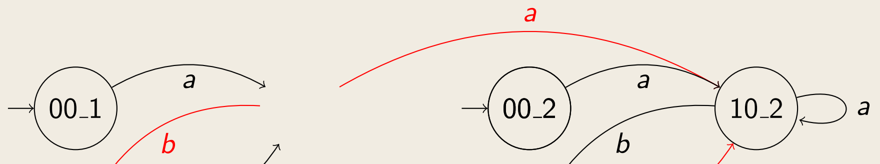
$a(a + ba)^\omega$:



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

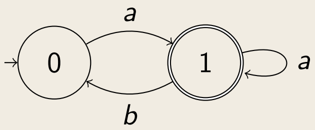


(i) Product Automaton(ii) Reachable States(iii) Clone(iv) Initial States Restricted to First Copy(v) Final States Restricted to First Atomaton of First Copy(vi) Ensure Acceptance in Both Copies $1 \to 2$**(vii) Ensure Acceptance in Both Copies** $2 \to 1$**(viii) Transitions of Product Automaton**

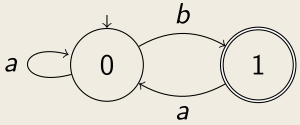


Explicit Construction of Intersection Automaton

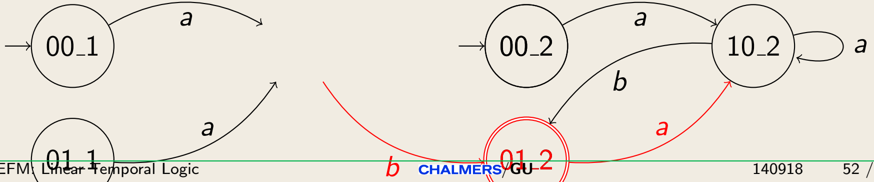
$a(a + ba)^\omega$:



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

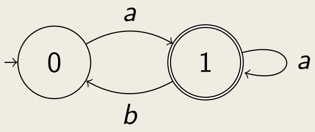


(i) Product Automaton(ii) Reachable States(iii) Clone(iv) Initial States Restricted to First Copy(v) Final States Restricted to First Atomaton of First Copy(vi) Ensure Acceptance in Both Copies $1 \rightarrow 2$**(vii) Ensure Acceptance in Both Copies** $2 \rightarrow 1$**(viii) Transitions of Product Automaton**

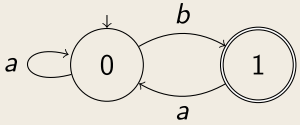


Explicit Construction of Intersection Automaton

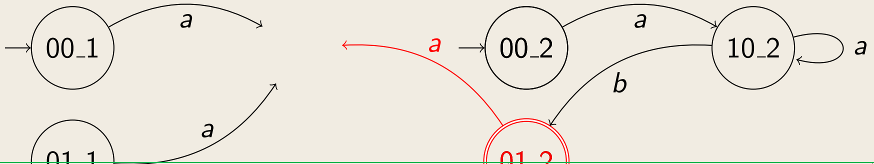
$a(a+ba)^\omega$:



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



(i) Product Automaton(ii) Reachable States(iii) Clone(iv) Initial States Restricted to First Copy(v) Final States Restricted to First Atomaton of First Copy(vi) Ensure Acceptance in Both Copies $1 \to 2$**(vii) Ensure Acceptance in Both Copies** $2 \to 1$**(viii) Transitions of Product Automaton**



Appendix II:

Construction of a Büchi Automaton $\mathcal{B}_\phi$ for an LTL-Formula ϕ

The General Case: Generalised Büchi Automata

A generalised Büchi automaton is defined as:

$$\mathcal{B}^g = (Q, \delta, I, \mathbb{F})$$

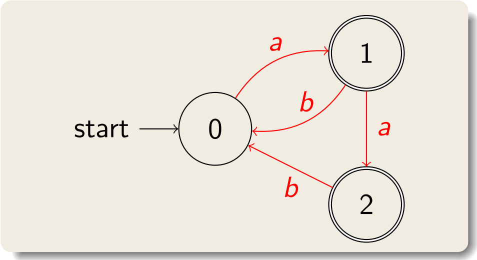
Q, δ, I as for standard Büchi automata

$\mathbb{F} = \{\mathcal{F}_1, \dots, \mathcal{F}_n\}$, where $\mathcal{F}_i = \{q_{i1}, \dots, q_{im_i}\} \subseteq Q$

Definition (Acceptance for generalised Büchi automata)

A generalised Büchi automaton accepts an ω-word $w \in \Sigma^\omega$ iff for every $i \in \{1, \dots, n\}$ at least one $q_{ik} \in \mathcal{F}_i$ is visited infinitely often.

Normal vs. Generalised Büchi Automata: Example



$\mathcal{B}^{normal}$ with $\mathcal{F} = \{1, 2\}$, $\quad$ $\mathcal{B}^{general}$ with $\mathbb{F} = \{ \overbrace{\{1\}}^{\mathcal{F}_1}, \overbrace{\{2\}}^{\mathcal{F}_2} \}$

Which ω-word is accepted by which automaton?

ω-word	$\mathcal{B}^{normal}$	$\mathcal{B}^{general}$
$(ab)^\omega$	✔	✘
$(aab)^\omega$	✔	✔

Fischer-Ladner Closure

Fischer-Ladner closure of an LTL-formula ϕ

$$FL(\phi) = \{\varphi | \varphi \text{ is subformula or negated subformula of } \phi\}$$

($\neg\neg\varphi$ is identified with φ)

Example

$$FL(r\,\mathcal{U}s) = \{r, \neg r, s, \neg s, r\,\mathcal{U}s, \neg(r\,\mathcal{U}s)\}$$

$\mathcal{B}_\phi$-Construction: Locations

Assumption:

$\mathcal{U}$ only temporal logic operator in LTL-formula (can express $\Box, \Diamond$ with $\mathcal{U}$)

Locations of $\mathcal{B}_\phi$ are $Q \subseteq 2^{FL(\phi)}$ where each $q \in Q$ satisfies:

Consistent, Total
- $\psi \in FL(\phi)$: exactly one of ψ and $\neg\psi$ in q
- $\psi_1 \mathcal{U} \psi_2 \in (FL(\phi) \backslash q)$ then $\psi_2 \notin q$

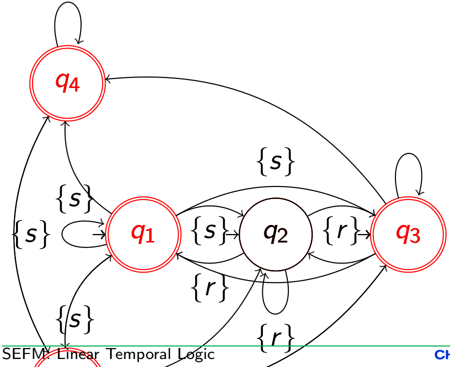
Downward Closed
- $\psi_1 \wedge \psi_2 \in q$: $\psi_1 \in q$ and $\psi_2 \in q$
- ...other propositional connectives similar
- $\psi_1 \mathcal{U} \psi_2 \in q$ then $\psi_1 \in q$ or $\psi_2 \in q$

$$FL(r\mathcal{U}s) = \{r, \neg r, s, \neg s, r\mathcal{U}s, \neg(r\mathcal{U}s)\}$$

	$\in Q$
$\{r\mathcal{U}s, \neg r, s\}$	✔
$\{r\mathcal{U}s, \neg r, \neg s\}$	✘
$\{\neg(r\mathcal{U}s), r, s\}$	✘
$\{\neg(r\mathcal{U}s), r, \neg s\}$	✔

$\mathcal{B}_\phi$-Construction: Transitions

$$\underbrace{\{r\,\mathcal{U}s, \neg r, s\}}_{q_1}, \underbrace{\{r\,\mathcal{U}s, r, \neg s\}}_{q_2}, \underbrace{\{r\,\mathcal{U}s, r, s\}}_{q_3}, \underbrace{\{\neg(r\,\mathcal{U}s), r, \neg s\}}_{q_4}, \underbrace{\{\neg(r\,\mathcal{U}s), \neg r, \neg s\}}_{q_5}$$



Transitions $(q, \alpha, q') \in \delta_\phi$:

$\alpha = q \cap \mathcal{P}$

$\mathcal{P}$ set of propositional variables outgoing edges of q_1 labeled $\{s\}$, of q_2 labeled $\{r\}$, etc.

1. If $\psi_1\,\mathcal{U}\psi_2 \in q$ and $\psi_2 \notin q$ then $\psi_1\,\mathcal{U}\psi_2 \in q'$
2. If $\psi_1\,\mathcal{U}\psi_2 \in (FL(\phi)\backslash q)$ and $\psi_1 \in q$ then $\psi_1\,\mathcal{U}\psi_2 \notin q'$

Initial locations

$$q \in I_\phi \text{ iff } \phi \in q$$

Accepting locations

Remarks on Generalized Büchi Automata

- Construction always gives exponential number of states in $|\phi|$
- Satisfiability checking of LTL is PSPACE-complete
- There exist (more complex) constructions that minimize number of required states
  - One of these is used in SPIN, which moreover computes the states lazily