Formal Methods – An Appetizer

Chapter 6: Model Checking

Flemming Nielson, Hanne Riis Nielson:
Formal Methods – An Appetizer.
ISBN 9783030051556, Springer 2019.

FormalMethods.dk

Techniques for Reasoning About Properties

Program Verification

We specify the properties holding at the nodes of the program graph using general predicates.

We obtain proof obligations that must be verified for each program graph.

Program Analysis

The analysis works with approximations and automatically computes the properties holding at the nodes of the program graph.

The correctness is established once for all when the analysis is specified.

Model Checking

The properties are formulated in a restricted logic and express reachability in a transition system.

The correctness follows from the relationship between the logic and the transition system.

Compared to program verification, model checking is less expressive but it can be fully automated.

Compared to program analysis, model checking is more expressive but its realisation is usually more costly.

Model Checking Works on Transition Systems

Program Graphs and Memories

- Program graphs represent the *control structure* of programs.
- Memories represent the *data structures* over which the program operates.
- Configurations represent the *state* in which we are at any point in time.
- The semantics defines *execution steps* (or *transitions*) between configurations

Transition Systems

States of transition systems combine program graphs and memories, but there is not necessarily a clear distinction between what is control and what is data.

6.1 Transition Systems

Flemming Nielson, Hanne Riis Nielson:
Formal Methods – An Appetizer.
ISBN 9783030051556, Springer 2019.

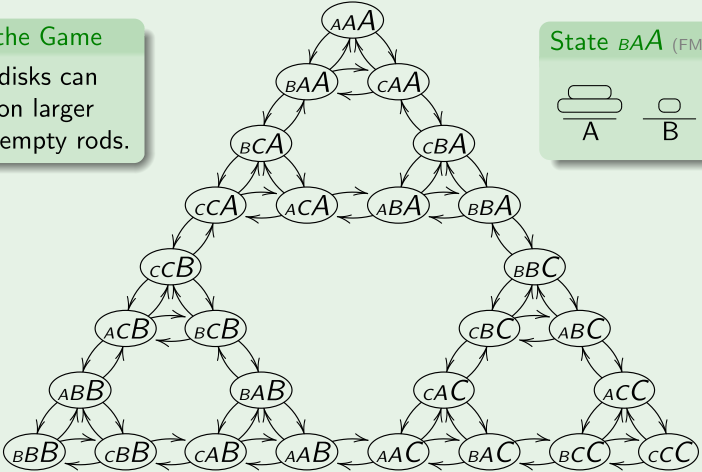
`FormalMethods.dk`

Example: Towers of Hanoi with Three Disks

Transition System (FM p 78)

Rule of the Game

Smaller disks can only go on larger disks or empty rods.



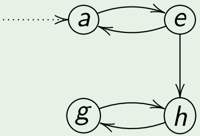
State BAA (FM p 78)

A B C

Transition Systems

Transition System

(FM p 80)



Labelling function

$\mathbf{L}(a) = \{v\}$
$\mathbf{L}(e) = \{v\}$
$\mathbf{L}(g) = \{c\}$
$\mathbf{L}(h) = \{c\}$

Transition System (FM p 77)

A *transition system* **TS** consists of

- **S**: a non-empty set of *states*
  $\{a, e, g, h\}$
- $\mathbf{I} \subseteq \mathbf{S}$: a non-empty set of *initial states*
  $\{a\}$
- $\longrightarrow \subseteq \mathbf{S} \times \mathbf{S}$: the *transition relation*;
  we write $\varsigma \longrightarrow \varsigma'$ when $(\varsigma, \varsigma') \in \longrightarrow$
- **AP**: a set of *atomic propositions*
  $\{v, c\}$
- $\mathbf{L} : \mathbf{S} \rightarrow \mathrm{PowerSet}(\mathbf{AP})$: a *labelling function*

For Towers of Hanoi (FM p 78)

$\mathbf{S} = \{{\scriptscriptstyle A}{\scriptstyle A}A, \cdots, {\scriptscriptstyle A}{\scriptstyle B}C, \cdots, {\scriptscriptstyle C}{\scriptstyle C}C\}$, $\mathbf{I} = \{{\scriptscriptstyle A}{\scriptstyle A}A\}$, $\mathbf{AP} = \mathbf{S}$ and $\mathbf{L}(\varsigma) = \{\varsigma\}$ for all $\varsigma \in \mathbf{S}$.

Paths and Reachability in Transition Systems

Paths (FM p 79)

A path is a (possible infinite) sequence of states $\varsigma_0 \varsigma_1 \cdots \varsigma_{n-1} \varsigma_n \cdots$ where $\varsigma_{n-1} \longrightarrow \varsigma_n$ for all n.

Paths are always as long as possible.
A path is only finite if it ends in a stuck state.
Path fragments need not be as long as possible.

Notation (FM p 79)

- **Path**(ς_0): the set of paths starting in ς_0
- **Reach**$_1(S_0)$: the set of states reachable from some state in S_0 in exactly one step
- **Reach**(S_0): the set of states reachable from some state in S_0 in zero or more steps

Tower's of Hanoi (FM p 79, 79)

- Are there any stuck states?
- Are there any infinite paths?
- Are there any finite paths?
- Trace the shortest path fragment from ${}_{A}AA$ to ${}_{C}CC$.
- Which states are reachable from **I**?
- Which states are not reachable from **I**?
- What is **Reach**$_1({}_{A}BC)$?
- What is **Reach**$({}_{A}BC)$?

6.2 Computation Tree Logic CTL

Flemming Nielson, Hanne Riis Nielson:
Formal Methods – An Appetizer.
ISBN 9783030051556, Springer 2019.

FormalMethods.dk

CTL Properties Φ: Reachability in one step

Atomic Proposition: ap

The proposition holds in a state if it is a label of the state, that is, ap $\in \mathbf{L}(\varsigma)$.

Reachability in one step: EX Φ

It is possible that in the next step we reach a state with the property Φ.

Reachability in one step: AX Φ

It is always the case that in the next step we reach a state with the property Φ.

$\varsigma \models \Phi$ means that Φ holds in state ς.

$e \models v$

$e \models \mathrm{EX}\, c$

$a \models \mathrm{AX}\, v$

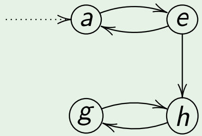
Try It Out

Do these formulae hold in a?

- $\mathrm{AX}\,(\mathrm{EX}\, c)$
- $\mathrm{EX}\,(\mathrm{AX}\, c)$

Transition System

(FM p 80)



Labelling function

$$\mathbf{L}(a) = \{v\}$$
$$\mathbf{L}(e) = \{v\}$$
$$\mathbf{L}(g) = \{c\}$$
$$\mathbf{L}(h) = \{c\}$$

CTL Properties Φ: Reachability

Reachability: EF Φ

It is possible that we after some steps will reach a state with the property Φ.

Reachability: AF Φ

It is always the case that after some steps will reach a state with the property Φ.

Logical Operators

On top of the CTL operators we can also use operators as $\neg\Phi$, $\Phi_1 \wedge \Phi_2$, $\Phi_1 \vee \Phi_2$, $\cdots$

$a \models \mathsf{EF}\, c$
$g \not\models \mathsf{EF}\, v$

$g \models \mathsf{AF}\, c$
$a \not\models \mathsf{AF}\, c$

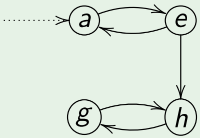
Try It Out

Do these formulae hold in a?

- $\mathsf{AF}\,(\mathsf{EX}\, c)$
- $\mathsf{EF}\,(\mathsf{AF}\, v)$
- $\mathsf{EF}\,(\neg\mathsf{AF}\, v)$

Transition System

(FM p 80)



Labelling function

$$\begin{aligned} \mathbf{L}(a) &= \{v\} \\ \mathbf{L}(e) &= \{v\} \\ \mathbf{L}(g) &= \{c\} \\ \mathbf{L}(h) &= \{c\} \end{aligned}$$

CTL Properties Φ: Unavoidability

Unavoidability: $\mathsf{EG}\,\Phi$

It is possible to select the steps such that all the states we reach have the property Φ.

Unavoidability: $\mathsf{AG}\,\Phi$

It is always the case that that all the states we reach have the property Φ.

$a \models \mathsf{EG}\, v$

$g \models \mathsf{AG}\, c$

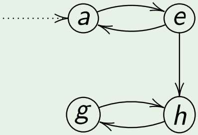
Try It Out

Do these formulae hold in a?

- $\mathsf{AX}\,(\mathsf{EG}\, c)$
- $\mathsf{EX}\,(\mathsf{AG}\, c)$
- $\mathsf{AG}\,(\neg \mathsf{EX}\, v)$

Transition System

(FM p 80)



Labelling function

$\mathbf{L}(a) = \{v\}$
$\mathbf{L}(e) = \{v\}$
$\mathbf{L}(g) = \{c\}$
$\mathbf{L}(h) = \{c\}$

Summary

	X one step	F some path	G unavoidable
E possible	$\mathsf{EX}\,\Phi$	$\mathsf{EF}\,\Phi$	$\mathsf{EG}\,\Phi$
A always	$\mathsf{AX}\,\Phi$	$\mathsf{AF}\,\Phi$	$\mathsf{AG}\,\Phi$

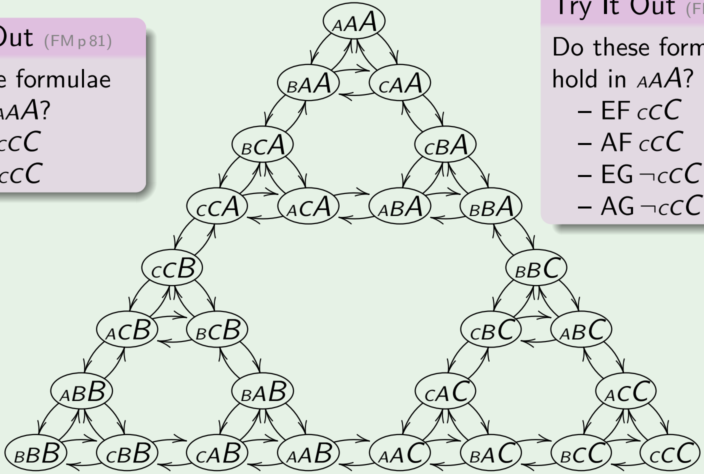
Example: Towers of Hanoi with Three Disks

Transition System (FM p 78)

Try It Out (FM p 81)

Do these formulae hold in ${}_{A}AA$?

- EX ${}_{C}CC$
- AX ${}_{C}CC$



Try It Out (FM p 81, 82)

Do these formulae hold in ${}_{A}AA$?

- EF ${}_{C}CC$
- AF ${}_{C}CC$
- EG $\neg {}_{C}CC$
- AG $\neg {}_{C}CC$

6.3 Syntax and Semantics of CTL

Flemming Nielson, Hanne Riis Nielson:
Formal Methods – An Appetizer.
ISBN 9783030051556, Springer 2019.

FormalMethods.dk

Syntax and Semantics of CTL

State Formulae (FM p 82)

$$\Phi \quad ::= \quad ap \mid \mathsf{E}\Psi \mid \mathsf{A}\Psi$$
$$\mid \quad \neg\Phi \mid \Phi_1 \wedge \Phi_2 \mid \cdots$$

Semantics $\varsigma \models \Phi$ (FM p 83)

The truth value is determined in a state ς:

$\varsigma \models ap$ iff $ap \in \mathbf{L}(\varsigma)$

$\varsigma \models \mathsf{E}\Psi$ iff $\exists \pi \in \mathbf{Path}(\varsigma) : \pi \models \Psi$

$\varsigma \models \mathsf{A}\Psi$ iff $\forall \pi \in \mathbf{Path}(\varsigma) : \pi \models \Psi$

$\varsigma \models \neg\Phi$ iff $\varsigma \not\models \Phi$

$\varsigma \models \Phi_1 \wedge \Phi_2$ iff $\varsigma \models \Phi_1 \wedge \varsigma \models \Phi_2$

$\vdots$

Path Formulae (FM p 82)

$$\Psi \quad ::= \quad \mathsf{X}\,\Phi \mid \mathsf{F}\,\Phi \mid \mathsf{G}\,\Phi \mid \Phi_1\,\mathsf{U}\,\Phi_2$$

Semantics $\pi \models \Psi$ (FM p 83)

The truth value is determined for a path $\pi = \varsigma_0\varsigma_1 \cdots \varsigma_n \cdots$

$\pi \models \mathsf{X}\,\Phi$ iff $\varsigma_1 \models \Phi \wedge n > 0$

$\pi \models \mathsf{F}\,\Phi$ iff $\exists n \geq 0 : \varsigma_n \models \Phi$

$\pi \models \mathsf{G}\,\Phi$ iff $\forall m : \varsigma_m \models \Phi$

$\pi \models \Phi_1\,\mathsf{U}\,\Phi_2$ iff $\exists n \geq 0 : \varsigma_n \models \Phi_2 \wedge \forall m < n : \varsigma_m \models \Phi_1$

So $\Phi_1\,\mathsf{U}\,\Phi_2$ means that Φ_1 holds on all states on the path up to where Φ_2 holds.

6.4 From Program Graphs to Transition Systems

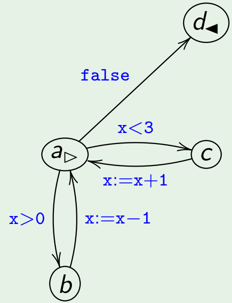
Flemming Nielson, Hanne Riis Nielson:
Formal Methods – An Appetizer.
ISBN 9783030051556, Springer 2019.

FormalMethods.dk

Model Checking For Program Graphs

Idea: A program graph and its semantics give rise to a transition system; we can use CTL properties to express properties of the system.

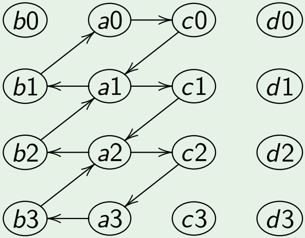
Program Graph (FM p 84)



$AG\,(\neg @_d)$

Transition System (FM p 85)

A state records a node and the value of x:



$(@_c \wedge \#_2) \Rightarrow AX\,(@_a \wedge \#_3)$

Atomic Propositions

$$\mathbf{AP} = \{@_a, @_b, @_c, @_d, \#_0, \#_1, \#_2, \#_3, \triangleright, \blacktriangleleft\}$$

Labelling function

(FM p 85)

$$\mathbf{L}(an) = \{@_a, \#_n, \triangleright\}$$
$$\mathbf{L}(bn) = \{@_b, \#_n\}$$
$$\mathbf{L}(cn) = \{@_c, \#_n\}$$
$$\mathbf{L}(dn) = \{@_d, \#_n, \blacktriangleleft\}$$

From Program Graphs To Transition Systems

The General Approach (FM p 84)

Given a program graph and its semantics:

- **S** is the set of configurations $\langle q; \sigma \rangle$
- **I** is the set of configurations of the form $\langle q_\triangleright; \sigma \rangle$
- we have $\langle q; \sigma \rangle \longrightarrow \langle q'; \sigma' \rangle$ if the semantics has $\langle q; \sigma \rangle \overset{\alpha}{\Longrightarrow} \langle q'; \sigma' \rangle$
- **AP** has three kinds of propositions:
  - $@_q$ for each node q
  - $\#_\sigma$ for each memory σ
  - $\triangleright$ and $\blacktriangleleft$
- $\mathbf{L}(\langle q; \sigma \rangle) = \begin{cases} \{@_q, \#_\sigma, \triangleright\} & \text{if } q = q_\triangleright \\ \{@_q, \#_\sigma, \blacktriangleleft\} & \text{if } q = q_\blacktriangleleft \\ \{@_q, \#_\sigma\} & \text{otherwise} \end{cases}$

Transition System (FM p 77)

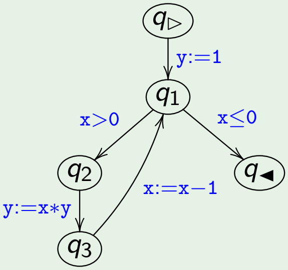
Recall that a *transition system* consists of

- **S**: set of states
- $\mathbf{I} \subseteq \mathbf{S}$: initial states
- $\longrightarrow \subseteq \mathbf{S} \times \mathbf{S}$: transition relation
- **AP**: *atomic propositions*
- $\mathbf{L} : \mathbf{S} \rightarrow \text{PowerSet}(\mathbf{AP})$: labelling function

Obs: The transition system may be infinite!

Model Checking The Factorial Program

Factorial Program (FM p 86)



- $\triangleright \Rightarrow \mathsf{EF} \blacktriangleleft$ expresses that when we start running the program there is a way for it to finish.
- $\triangleright \Rightarrow \mathsf{AF} \blacktriangleleft$ expresses that when we start running the program it will eventually finish.
- $(\triangleright \wedge \#_{[\mathtt{x}:7,\,\mathtt{y}:0]}) \Rightarrow \mathsf{AF}\,(\blacktriangleleft \wedge \#_{[\mathtt{x}:0,\,\mathtt{y}:5040]})$ expresses that the program will compute the factorial of 7 to be 5040.

Try It Out (FM p 86)

Let us limit the values of x and y to be in the sets $\{0, 1, 2, 3\}$ and $\{0, 1, 2, 3, 4, 5, 6\}$, resp.

How many states will there be in the transition system constructed from the program graph?

Determine whether or not the following formulae hold in the resulting transition system:

- $\triangleright \Rightarrow \mathsf{EF} \blacktriangleleft$
- $\triangleright \Rightarrow \mathsf{AF} \blacktriangleleft$

6.5 Towards an Algorithm

Flemming Nielson, Hanne Riis Nielson:
Formal Methods – An Appetizer.
ISBN 9783030051556, Springer 2019.

`FormalMethods.dk`

Does The CTL Property Φ Hold?

Idea: We devise algorithms computing the set of states where Φ hold

Some Easy Cases (FM p 87)

$$\mathbf{Sat}(\mathsf{ap}) = \{\varsigma \in \mathbf{S} \mid \mathsf{ap} \in \mathbf{L}(\varsigma)\}$$
$$\mathbf{Sat}(\Phi_1 \land \Phi_2) = \mathbf{Sat}(\Phi_1) \cap \mathbf{Sat}(\Phi_2)$$
$$\mathbf{Sat}(\neg\Phi) = \mathbf{S} \setminus \mathbf{Sat}(\Phi)$$

$$\mathbf{Sat}(\Phi) = \{\varsigma \mid \varsigma \models \Phi\}$$

Some Simpler Cases (FM p 87, 87)

$$\mathbf{Sat}(\mathsf{EX}\,\Phi) = \{\varsigma \mid \mathbf{Reach}_1(\{\varsigma\}) \cap \mathbf{Sat}(\Phi) \neq \{\,\}\}$$
$$\mathbf{Sat}(\mathsf{AX}\,\Phi) = \{\varsigma \mid \mathbf{Reach}_1(\{\varsigma\}) \subseteq \mathbf{Sat}(\Phi)\}$$

Recall (FM p 79)

$\mathbf{Reach}_1(\{\varsigma\})$ is the set of states reachable from ς in exactly one step

$$\mathbf{Sat}(\mathsf{EF}\,\Phi) = \{\varsigma \mid \mathbf{Reach}(\{\varsigma\}) \cap \mathbf{Sat}(\Phi) \neq \{\,\}\}$$
$$\mathbf{Sat}(\mathsf{AG}\,\Phi) = \{\varsigma \mid \mathbf{Reach}(\{\varsigma\}) \subseteq \mathbf{Sat}(\Phi)\}$$

$\mathbf{Reach}(\{\varsigma\})$ is the set of states reachable from ς in zero or more steps

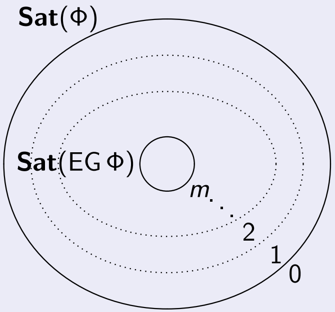
The More Complex Cases

Algorithm For EG Φ (FM p 88)

$S := \mathbf{Sat}(\Phi)$
while possible do
  choose $\varsigma \in S$
    if $\mathbf{Reach}_1(\{\varsigma\}) \cap S = \{\,\}$
    then $S := S \setminus \{\varsigma\}$

Recall: EG Φ holds if it is possible to select the steps such that all the states we reach have the property Φ.

Computing EG Φ (FM p 88)



Algorithm For AF Φ (FM p 88)

We exploit that AF Φ is equivalent to $\neg(\mathrm{EG}\,(\neg\Phi))$ (under appropriate assumptions)

The Remaining Cases (FM p 90)

Efficient algorithms also exist for $\mathrm{E}(\Phi_1 \,\mathrm{U}\, \Phi_2)$ and $\mathrm{A}(\Phi_1 \,\mathrm{U}\, \Phi_2)$