

# Algorithms for Model Checking (2IW55)

## Lecture 7

### The $\mu$ -Calculus

(Chapter 7 in *Model Checking* by Clarke, Grumberg & Peled)

Tim Willemse

(timw@win.tue.nl)

<http://www.win.tue.nl/~timw>

HG 6.76

$\mu$ -Calculus: syntax and semantics

Examples

Complexity

Emerson-Lei Algorithm

Embedding CTL-formulae

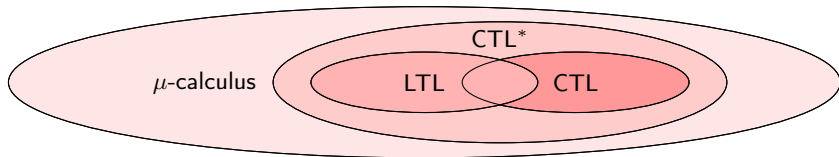
Conclusions

Exercise

Recall: symbolic model checking for CTL was based on **fixed points**.

Idea of  $\mu$ -calculus: add fixed point operators as primitives to basic modal logic.

- ▶  $\mu$ -calculus is **very** expressive (subsumes CTL, LTL, CTL\*).
- ▶  $\mu$ -calculus is very **pure** (“assembly language” for modal logic, cf:  $\lambda$ -calculus for functional programming).
- ▶ drawback: lack of intuition.
- ▶ fragments of the  $\mu$ -calculus are the basis for practical model checkers, such as  $\mu$ CRL, mCRL2, CADP, Concurrency Workbench.



## Kripke Structures and Labelled Transition Systems

Mix of Kripke Systems and Labelled Transition Systems:  $M = \langle S, Act, R, L \rangle$  over a set  $AP$  of atomic propositions:

- ▶  $S$  is a set of states
- ▶  $Act$  is a set of action labels
- ▶  $R$  is a **labelled** transition relation:  $R \subseteq S \times Act \times S$
- ▶  $L$  is a labelling:  $L \in S \rightarrow 2^{AP}$

**Notation:**  $s \xrightarrow{a} t$  denotes  $(s, a, t) \in R$

Special cases:

- ▶ Kripke Structures:  $Act$  is a singleton (only one transition relation)
- ▶ LTS (process algebra):  $AP$  is empty (only propositions true and false)

Let the following sets be given:

- ▶  $AP$  (atomic propositions),
- ▶  $Act$  (action labels) and
- ▶  $Var$  (formal variables).

The syntax of  $\mu$ -calculus formulae  $f, g$  is defined by the following grammar:

$$f, g ::= \text{true} \mid p \mid X \mid \neg f \mid f \wedge g \mid [a]f \mid \nu X.f$$

Note:

- ▶  $p \in AP, X \in Var, a \in Act$ .
- ▶  $[a]f$  means “for all **direct**  $a$ -successors,  $f$  holds”.

## Some notation and terminology:

- ▶ An occurrence of  $X$  is **bound** by a surrounding fixed point symbol  $\nu X$ . Unbound occurrences of  $X$  are called **free**.
- ▶ A formula is **closed** if it has no free variables, otherwise it is called **open**
- ▶ An **environment**  $e$  interprets the free formal variables  $X$  as a set of states
  - Mixed Kripke Structure  $M = \langle S, Act, R, L \rangle$
  - $e : Var \rightarrow 2^S$
  - $e[X := V]$  is an environment like  $e$ , but  $X$  is set to  $V$ :

$$e[X := V](Y) := \begin{cases} V & \text{if } Y = X \\ e(Y) & \text{otherwise} \end{cases}$$

- ▶ The **semantics** of a formula  $f$  is a **set of states** of a Mixed Kripke Structure

Fix a system:  $M = \langle S, Act, R, L \rangle$

- ▶  $[f]_e$  denotes the set of states where  $f$  holds given context  $e : Var \rightarrow 2^S$ :

$$[\text{true}]_e = S$$

$$[p]_e = \{s \mid p \in L(s)\}$$

$$[X]_e = e(X)$$

$$[\neg f]_e = S \setminus [f]_e$$

$$[f \wedge g]_e = [f]_e \cap [g]_e$$

$$[[a]f]_e = \{s \mid \forall t. s \xrightarrow{a} t \Rightarrow t \in [f]_e\}$$

$$[\nu X.f]_e = \text{gfp}(Z \mapsto [f]_{e[X:=Z]})$$

Fix a system:  $M = \langle S, Act, R, L \rangle$

- ▶  $[f]_e$  denotes the set of states where  $f$  holds given context  $e : Var \rightarrow 2^S$ :

$$\begin{aligned} [\text{true}]_e &= S \\ [p]_e &= \{s \mid p \in L(s)\} \\ [X]_e &= e(X) \end{aligned}$$

$$\begin{aligned} [\neg f]_e &= S \setminus [f]_e \\ [f \wedge g]_e &= [f]_e \cap [g]_e \\ [[a]f]_e &= \{s \mid \forall t. s \xrightarrow{a} t \Rightarrow t \in [f]_e\} \\ [\nu X.f]_e &= \text{gfp}(Z \mapsto [f]_{e[X:=Z]}) \end{aligned}$$

- ▶  $[\nu X.f]_e$  requires monotonicity of  $[f]_{e[X:=Z]}$ .
- ▶ **Syntactic Monotonicity Criterion:** monotonicity is guaranteed if, in  $\nu X.f$ , formal variable  $X$  occurs under an **even** number of negations ( $\neg$ ) in  $f$ .



Fix a system:  $M = \langle S, Act, R, L \rangle$

- ▶  $[f]_e$  denotes the set of states where  $f$  holds given context  $e : Var \rightarrow 2^S$ :

$$\begin{aligned}[true]_e &= S \\ [p]_e &= \{s \mid p \in L(s)\} \\ [X]_e &= e(X)\end{aligned}$$

$$\begin{aligned}[\neg f]_e &= S \setminus [f]_e \\ [f \wedge g]_e &= [f]_e \cap [g]_e \\ [[a]f]_e &= \{s \mid \forall t. s \xrightarrow{a} t \Rightarrow t \in [f]_e\} \\ [\nu X.f]_e &= gfp(Z \mapsto [f]_{e[X:=Z]})\end{aligned}$$

- ▶  $[\nu X.f]_e$  requires monotonicity of  $[f]_{e[X:=Z]}$ .
- ▶ **Syntactic Monotonicity Criterion:** monotonicity is guaranteed if, in  $\nu X.f$ , formal variable  $X$  occurs under an **even** number of negations ( $\neg$ ) in  $f$ .

The semantics immediately gives rise to a **naive algorithm** for model checking  $\mu$ -calculus (compute  $gfp$  by iteration).

- ▶ Extend the grammar with the following **shorthands** with semantics:

$\text{false} := \neg \text{true}$ $f \vee g := \neg((\neg f) \wedge (\neg g))$	$[\text{false}]_e = \emptyset$ $[f \vee g]_e = [f]_e \cup [g]_e$
$\langle a \rangle f := \neg([a](\neg f))$	$[\langle a \rangle f]_e = \{s \mid \exists t. s \xrightarrow{a} t \wedge t \in [f]_e\}$
$\mu X.f := \neg(\nu X. \neg f[X := \neg X])$	$[\mu X.f]_e = \text{lfp}(Z \mapsto [f]_{e[X:=Z]})$

- ▶ A  $\mu$ -calculus formula is in **positive normal form** if negations occur only in front of propositions.
- ▶ Transform a formula into positive normal form by driving negations inward.
- ▶ Syntactic monotonicity prevents single negations in front of formal variables.

$\mu$ -Calculus: syntax and semantics

Examples

Complexity

Emerson-Lei Algorithm

Embedding CTL-formulae

Conclusions

Exercise

- ▶ Not a syntactically monotone  $\mu$ -calculus formula:  $\mu X. \neg X$
- ▶ Semantically monotone **but** not syntactically monotone:  $\mu X. \neg X \vee \text{true}$
- ▶ Let  $\text{Act} = \{a\}$ :
  - E G  $f \dots \nu X. f \wedge \langle a \rangle X$
  - E  $[f \text{ U } g] \dots \mu X. g \vee (f \wedge \langle a \rangle X)$
  - Every  $p$  is inevitably followed by a  $q$ :  $\nu X_1. \left( (p \Rightarrow (\mu X_2. q \vee [a]X_2)) \wedge [a]X_1 \right)$
- ▶ **Special case:**  $X_1$  does not occur within the scope of  $\mu X_2$ .
- ▶ The last formula can therefore be evaluated “inside-out”:

$$\begin{array}{lcl}
 X_2^0 & = & \text{false} \\
 X_2^1 & = & q \vee [a]X_2^0 \\
 X_2^2 & = & q \vee [a]X_2^1 \\
 \dots & X_2^\omega & = & q \vee [a]X_2^\omega
 \end{array}
 \quad \Longrightarrow \quad
 \begin{array}{lcl}
 X_1^0 & = & \text{true} \\
 X_1^1 & = & (p \Rightarrow X_2^\omega) \wedge [a]X_1^0 \\
 X_1^2 & = & (p \Rightarrow X_2^\omega) \wedge [a]X_1^1 \\
 \dots & X_1^\omega & = & (p \Rightarrow X_2^\omega) \wedge [a]X_1^\omega
 \end{array}$$

## A more difficult case

- ▶ On some path,  $h$  holds infinitely often:  $\nu X_1. \langle a \rangle (\mu X_2. (X_1 \wedge h) \vee \langle a \rangle X_2)$
- ▶ **Problem:** the inner fixed point depends crucially on  $X_1$ .

$$\begin{array}{rcl}
 X_1^0 & = & \text{true} \\
 & & X_2^{00} = \text{false} \\
 & & X_2^{01} = (X_1^0 \wedge h) \vee \langle a \rangle X_2^{00} \\
 & & X_2^{02} = (X_1^0 \wedge h) \vee \langle a \rangle X_2^{01} \\
 & & X_2^{0\omega} = (X_1^0 \wedge h) \vee \langle a \rangle X_2^{0\omega} \\
 X_1^1 & = & \dots \\
 & & \langle a \rangle X_2^{0\omega} \\
 & & X_2^{10} = \text{false} \\
 & & X_2^{11} = (X_1^1 \wedge h) \vee \langle a \rangle X_2^{10} \\
 & & X_2^{1\omega} = (X_1^1 \wedge h) \vee \langle a \rangle X_2^{1\omega} \\
 X_1^2 & = & \dots \\
 & & \langle a \rangle X_2^{1\omega} \\
 \dots & X_1^\omega & = \langle a \rangle X_2^{\omega\omega}
 \end{array}$$

$\mu$ -Calculus: syntax and semantics

Examples

Complexity

Emerson-Lei Algorithm

Embedding CTL-formulae

Conclusions

Exercise

## Complexity of naive $\mu$ -Calculus algorithm

- ▶ We check formula  $f$  with at most  $k$  nested fixed points on the Kripke Structure  $M = \langle S, R, Act, L \rangle$ .
- ▶ In  $\nu X_1. \langle a \rangle (\mu X_2. (X_1 \wedge h) \vee \langle a \rangle X_2)$ :
  - The outermost (greatest) fixed point can decrease at most  $|S|$  times (recall that  $S$  is finite)
  - In total, the innermost fixed point of formula  $f$  is evaluated at most  $|S|^2$  times.
- ▶ In general: the innermost fixed point of formula  $f$  is evaluated at most  $|S|^k$  times.
- ▶ Each iteration requires up to  $|M| \times |f|$  steps.
- ▶ **Total time complexity** of naive algorithm:  $\mathcal{O}((|S| + |R|) \times |f| \times |S|^k)$ .

A more careful analysis will yield a more optimal treatment for nested fixed points **of the same type**.

The complexity of a  $\mu$ -calculus formula depends on the fixed points (*analogue*: the complexity of first-order formulae depends on the universal/existential quantifiers)

- ▶ **Basic idea:** find a syntactic complexity measure that approaches the semantic complexity
- ▶ **Nesting Depth:** maximum number of nested fixed points in a positive normal form

$ND(f)$	$:= 0$	for $f \in \{p, \neg p, X\}$
$ND(@f)$	$:= ND(f)$	for $@ \in \{[a], \langle a \rangle\}$
$ND(f \square g)$	$:= \max(ND(f), ND(g))$	for $\square \in \{\wedge, \vee\}$
$ND(\overset{\mu}{\nu} X.f)$	$:= 1 + ND(f)$	for $\overset{\mu}{\nu} \in \{\mu, \nu\}$

- ▶ Example:  $ND\left((\mu X_1. \nu X_2. X_1 \vee X_2) \wedge (\mu X_3. \mu X_4. (X_3 \wedge \mu X_5. p \vee X_5))\right)$



The complexity of a  $\mu$ -calculus formula depends on the fixed points (*analogue*: the complexity of first-order formulae depends on the universal/existential quantifiers)

- ▶ **Basic idea**: find a syntactic complexity measure that approaches the semantic complexity
- ▶ **Nesting Depth**: maximum number of nested fixed points in a positive normal form

$ND(f)$	$:= 0$	for $f \in \{p, \neg p, X\}$
$ND(@f)$	$:= ND(f)$	for $@ \in \{[a], \langle a \rangle\}$
$ND(f \square g)$	$:= \max(ND(f), ND(g))$	for $\square \in \{\wedge, \vee\}$
$ND(\overset{\mu}{\nu} X.f)$	$:= 1 + ND(f)$	for $\overset{\mu}{\nu} \in \{\mu, \nu\}$

- ▶ Example:  $ND\left((\mu X_1. \nu X_2. X_1 \vee X_2) \wedge (\mu X_3. \mu X_4. (X_3 \wedge \mu X_5. p \vee X_5))\right) = 3$
- ▶  $X_3, X_4$  and  $X_5$  have no alternation between fixed point signs

- ▶ Capture **alternation**
- ▶ **Alternation Depth**: number of **alternating** fixed points of a formula in positive normal form.

$AD(f) := 0$	for $f \in \{p, \neg p, X\}$
$AD(@f) := AD(f)$	for $@ \in \{[a], \langle a \rangle\}$
$AD(f \square g) := \max(AD(f), AD(g))$	for $\square \in \{\wedge, \vee\}$
$AD(\mu X.f) := 1 + \max\{AD(g) \mid g \text{ is a } \nu\text{-subformula of } f\}$	
$AD(\nu X.f) := 1 + \max\{AD(g) \mid g \text{ is a } \mu\text{-subformula of } f\}$	

- ▶ **Examples:**

$$AD\left((\mu X_1. \nu X_2. X_1 \vee X_2) \wedge (\mu X_3. \mu X_4. (X_3 \wedge \mu X_5. p \vee X_5))\right)$$

$$AD\left((\mu X_1. \nu X_2. X_1 \vee X_2) \wedge (\mu X_3. \nu X_4. (X_3 \wedge \mu X_5. p \vee X_5))\right)$$

- ▶ Capture **alternation**
- ▶ **Alternation Depth**: number of **alternating** fixed points of a formula in positive normal form.

$AD(f) := 0$	for $f \in \{p, \neg p, X\}$
$AD(@f) := AD(f)$	for $@ \in \{[a], \langle a \rangle\}$
$AD(f \square g) := \max(AD(f), AD(g))$	for $\square \in \{\wedge, \vee\}$
$AD(\mu X.f) := 1 + \max\{AD(g) \mid g \text{ is a } \nu\text{-subformula of } f\}$	
$AD(\nu X.f) := 1 + \max\{AD(g) \mid g \text{ is a } \mu\text{-subformula of } f\}$	

- ▶ **Examples:**

$$AD\left((\mu X_1. \nu X_2. X_1 \vee X_2) \wedge (\mu X_3. \mu X_4. (X_3 \wedge \mu X_5. p \vee X_5))\right) = 2$$

$$AD\left((\mu X_1. \nu X_2. X_1 \vee X_2) \wedge (\mu X_3. \nu X_4. (X_3 \wedge \mu X_5. p \vee X_5))\right) = 3$$

- ▶  $X_5$  does not depend on  $X_3$  and  $X_4$

- ▶ **Dependent Alternation Depth (dAD):** number of alternating fixed points, such that the innermost fixed point **depends** on the outermost.
- ▶ The definition of  $dAD$  is identical to  $AD$ , except for

$$\begin{aligned}
 dAD(\mu X.f) &:= \max(dAD(f), \\
 &\quad 1 + \max\{dAD(g) \mid \\
 &\quad \quad g \text{ is a } \nu\text{-subformula of } f \text{ and } X \text{ occurs in } g\}) \\
 dAD(\nu X.f) &:= \max(dAD(f), \\
 &\quad 1 + \max\{dAD(g) \mid \\
 &\quad \quad g \text{ is a } \mu\text{-subformula of } f \text{ and } X \text{ occurs in } g\})
 \end{aligned}$$

- ▶ **Examples:**

$$\begin{aligned}
 &dAD\left((\mu X_1. \nu X_2. X_1 \vee X_2) \wedge (\mu X_3. \mu X_4. (X_3 \wedge \mu X_5. p \vee X_5))\right) \\
 &dAD\left((\mu X_1. \nu X_2. X_1 \vee X_2) \wedge (\mu X_3. \nu X_4. (X_3 \wedge \mu X_5. p \vee X_5))\right)
 \end{aligned}$$

- ▶ **Dependent Alternation Depth (dAD):** number of alternating fixed points, such that the innermost fixed point **depends** on the outermost.
- ▶ The definition of  $dAD$  is identical to  $AD$ , except for

$$\begin{aligned}
 dAD(\mu X.f) &:= \max(dAD(f), \\
 &\quad 1 + \max\{dAD(g) \mid \\
 &\quad \quad g \text{ is a } \nu\text{-subformula of } f \text{ and } X \text{ occurs in } g\}) \\
 dAD(\nu X.f) &:= \max(dAD(f), \\
 &\quad 1 + \max\{dAD(g) \mid \\
 &\quad \quad g \text{ is a } \mu\text{-subformula of } f \text{ and } X \text{ occurs in } g\})
 \end{aligned}$$

- ▶ **Examples:**

$$dAD\left((\mu X_1. \nu X_2. X_1 \vee X_2) \wedge (\mu X_3. \mu X_4. (X_3 \wedge \mu X_5. p \vee X_5))\right) = 2$$

$$dAD\left((\mu X_1. \nu X_2. X_1 \vee X_2) \wedge (\mu X_3. \nu X_4. (X_3 \wedge \mu X_5. p \vee X_5))\right) = 2$$

$\mu$ -Calculus: syntax and semantics

Examples

Complexity

**Emerson-Lei Algorithm**

Embedding CTL-formulae

Conclusions

Exercise

- ▶ Given a **finite** set  $S$  and a **monotonic**  $\tau : 2^S \rightarrow 2^S$  in the partial order  $(2^S, \subseteq)$ .
- ▶ We used to compute the least fixed point from  $\emptyset$ :

$$\emptyset \subseteq \tau(\emptyset) \subseteq \tau^2(\emptyset) \subseteq \dots \subseteq \tau^i(\emptyset) = \tau^{i+1}(\emptyset)$$

then  $\mu X. \tau(X) = \tau^i(\emptyset)$

- ▶ Actually, instead of  $\emptyset$ , we can start in any set known to be **smaller** than the fixed point:

- ▶ Given a **finite** set  $S$  and a **monotonic**  $\tau : 2^S \rightarrow 2^S$  in the partial order  $(2^S, \subseteq)$ .
- ▶ We used to compute the least fixed point from  $\emptyset$ :

$$\emptyset \subseteq \tau(\emptyset) \subseteq \tau^2(\emptyset) \subseteq \dots \subseteq \tau^i(\emptyset) = \tau^{i+1}(\emptyset)$$

then  $\mu X. \tau(X) = \tau^i(\emptyset)$

- ▶ Actually, instead of  $\emptyset$ , we can start in any set known to be **smaller** than the fixed point:

- Assume  $W \subseteq \mu X. \tau(X)$ , so we have:

$$\emptyset \subseteq W \subseteq \tau^i(\emptyset)$$

- By monotonicity and the definition of fixed points:

$$\tau^i(\emptyset) \subseteq \tau^i(W) \subseteq \tau^{2i}(\emptyset) = \tau^i(\emptyset)$$

- So if  $W \subseteq \mu X. \tau(X)$  we compute the least fixed point as:

$$W, \tau(W), \tau^2(W), \dots, \tau^j(W) = \tau^{j+1}(W)$$

This converges at some  $j \leq i$  (may be  $j < i$ )



- ▶ The observations on the previous slide can speed up computations of nested fixed points.
- ▶ Consider two nested  $\mu$ -fixed points:  $\mu X_1. f(X_1, \mu X_2. g(X_1, X_2))$
- ▶ Start approximation of  $X_1$  and  $X_2$  with  $X_1^0 = X_2^0 = \text{false}$ :

$$\begin{array}{rcl}
 X_1^0 & = & \text{false} \\
 & & X_2^{00} = \text{false} \\
 & & X_2^{01} = g(X_1^0, X_2^{00}) \\
 & & \dots \quad X_2^{0\omega} = g(X_1^0, X_2^{0\omega}) \\
 X_1^1 & = & f(X_1^0, X_2^{0\omega})
 \end{array}$$

- ▶ Clearly,  $X_1^0 \subseteq X_1^1$ , so also  $X_2^{0\omega} = \mu X_2. g(X_1^0, X_2) \subseteq \mu X_2. g(X_1^1, X_2) = X_2^{1\omega}$ . So, approximating  $X_2$  can start at  $X_2^{0\omega}$  instead of at  $\text{false}$ :

$$\begin{array}{rcl}
 & & X_2^{10} = X_2^{0\omega} \\
 & & \dots \quad X_2^{1\omega} = g(X_1^1, X_2^{1\omega}) \\
 X_1^2 & = & f(X_1^1, X_2^{1\omega})
 \end{array}$$

Given:

- ▶ Mixed Kripke Structure:  $M = \langle S, R, Act, L \rangle$
- ▶ A  $\mu$ -Calculus formula  $f$  and an environment  $e$

Returns:  $[f]_e$ , the set of states in  $S$  where  $f$  holds.

Idea:

- ▶ The function  $\text{eval}(f)$  proceeds by recursion on  $f$ , using iteration for the fixed points.
- ▶ The value of the current approximation for variable  $X_i$  is **stored** in array  $A[i]$ , in order to reuse it in later iterations.
- ▶ **Reset**  $A[i]$  only if:
  - a higher  $X_j$  of **different** sign changed, and
  - $\overset{\mu}{\vee} X_i.f$  contains free variables.

Initialisation:

```
for all variables  $X_i$  do  
  if  $X_i$  is bound by a  $\mu$  then  $A[i] := \text{false}$ ;  
  else if  $X_i$  is bound by a  $\nu$  then  $A[i] := \text{true}$ ;  
  else  $A[i] := e(X_i)$   
  end if  
end for
```

```

function eval( $f$ )
  if  $f = X_i$  then return  $A[i]$ 
  else if  $f = g_1 \vee g_2$  then return eval( $g_1$ )  $\cup$  eval( $g_2$ )
  else if ... then ...
  else if  $f = \mu X_i.g(X_i)$  then
    if the surrounding binder of  $f$  is a  $\nu$  then
      for all open subformulae of  $f$  of the form  $\mu X_k.g$  do  $A[k] := false$ 
      end for
    end if
  repeat
     $X_{old} := A[i]$ ;
     $A[i] := eval(g)$ ;
  until  $A[i] = X_{old}$ 
  return  $A[i]$ 
  end if
end function

```

{continue from previous value}

Given a formula  $\nu X_1. \nu X_2. \mu X_3. \mu X_4. (X_1 \vee X_2 \vee (\mu X_5. X_5 \wedge p))$

- ▶ When computing  $\nu X_2$ ,  $\mu X_4$  and  $\mu X_5$ : no reset is needed because the surrounding binder has the same sign.
- ▶ When computing  $X_3$ :
  - Reset  $X_3, X_4$ : their subformula contains  $X_1$  and  $X_2$  as free variables
  - Do not reset  $X_5$ : the subformula  $(\mu X_5. X_5 \wedge p)$  is closed

Modifications with respect to the book (p. 105):

- ▶ We identified  $e$  and  $A[i]$  (they play the same role)
- ▶ The restriction to reset **open** formulae only makes the algorithm more efficient. This is essential for CTL (see later).
- ▶ The book has a slightly different algorithm (correctness unclear to me): we presented the original Emerson and Lei algorithm (1986).

## Complexity analysis

- ▶ Let formula  $f$  be given, with dependent alternation depth  $dAD(f) = d$ .
- ▶ Let the Kripke Structure be  $\langle S, Act, R, L \rangle$ .
- ▶ Take a block of fixed points of the same type:
  - its length is at most  $|f|$ .
  - the value of each fixed point in it can grow/shrink at most  $|S|$  times.
- ▶ In total, the innermost block will have no more than  $(|f| \cdot |S|)^d$  iterations of the repeat-loop.
- ▶ Each iteration requires time at most  $\mathcal{O}(|f| \cdot (|S| + |R|))$ .
- ▶ Hence: the overall complexity of the Emerson-Lei algorithm is  $\mathcal{O}(|f| \cdot (|S| + |R|) \cdot (|f| \cdot |S|)^d)$

$\mu$ -Calculus: syntax and semantics

Examples

Complexity

Emerson-Lei Algorithm

Embedding CTL-formulae

Conclusions

Exercise

Again, assume  $Act = \{a\}$ . Given the fixed point characterisation of CTL, there is a straightforward translation of CTL to the  $\mu$ -calculus:

- ▶  $Tr(p) = p$
- ▶  $Tr(\neg f) = \neg Tr(f)$
- ▶  $Tr(f \wedge g) = Tr(f) \wedge Tr(g)$
- ▶  $Tr(E X f) = \langle a \rangle Tr(f)$
- ▶  $Tr(E G f) = \nu Y. (Tr(f) \wedge \langle a \rangle Y)$
- ▶  $Tr(E [f U g]) = \mu Y. (Tr(g) \vee (Tr(f) \wedge \langle a \rangle Y))$

Note:

- ▶  $Tr(f)$  is syntactically monotone
- ▶  $Tr(f)$  is a closed  $\mu$ -calculus formula
- ▶  $dAD(Tr(f)) \leq 1$ , which is called the **alternation free** fragment of the  $\mu$ -calculus
- ▶  $AD(Tr(f))$  is not bounded!



$\mu$ -Calculus: syntax and semantics

Examples

Complexity

Emerson-Lei Algorithm

Embedding CTL-formulae

Conclusions

Exercise

- ▶ the  $\mu$ -calculus incorporates **least and greatest fixed points** directly in the logic.
- ▶ the **naive** algorithm is exponential in the nesting depth of fixed points.
- ▶ a careful analysis leads to an algorithm which is **exponential** in the **(dependent) alternation depth** only,
- ▶ Hence: alternation free  $\mu$ -calculus is **linear** in the Kripke Structure and **polynomial** in the formula.
- ▶ CTL translates into the **alternation free** fragment of the  $\mu$ -calculus.
- ▶ for the latter we essentially needed the **dependent** alternation depth.
- ▶ fairness constraints typically lead to one extra alternation ( $dAD(f) = 2$ )

$\mu$ -Calculus: syntax and semantics

Examples

Complexity

Emerson-Lei Algorithm

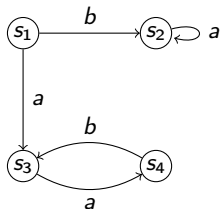
Embedding CTL-formulae

Conclusions

Exercise

Consider the following  $\mu$ -calculus formula  $\phi$  and LTS  $\mathcal{L}$ :

$$\phi := \nu X. \left( [a]X \wedge \nu Y. \mu Z. (\langle b \rangle Y \vee \langle a \rangle Z) \right)$$



- ▶ Compute the set of states where  $\phi$  holds with the naive algorithm (give all intermediate approximations).
- ▶ Compute the set of states where  $\phi$  holds with the Emerson-Lei's algorithm (give all intermediate approximations).
- ▶ Explain in natural language the meaning of formula  $\phi$ .