

Introducing Modal Fixed-Point Operators into CCSL

Christian Kissig

Univ. of Techn. Dresden, Dept. of CS

kissig@tcs.inf.tu-dresden.de

Introduction and Motivation

Introduction

● Introduction and Motivation

● The running Example

Preliminaries

Modal Operators

Modal Fixed-Point Operators

Conclusions

- Abstract Data Types / Classes modelled with Algebras
- Processes / Dynamic Systems are modelled with Coalgebras
- Modal Logic a natural Means to specify Dynamic Systems
- CCSL contains only infinitary Modal Operators
- CCSL does not support Finiteness Properties
- CCSL does not allow recursive Properties

Outline of the following 30min :

- Preliminaries & CCSL
- Introduction of the Modal Operators
- Introduction of the Modal Fixed-Point Operators

The running Example

Introduction

- Introduction and Motivation
- The running Example

Preliminaries

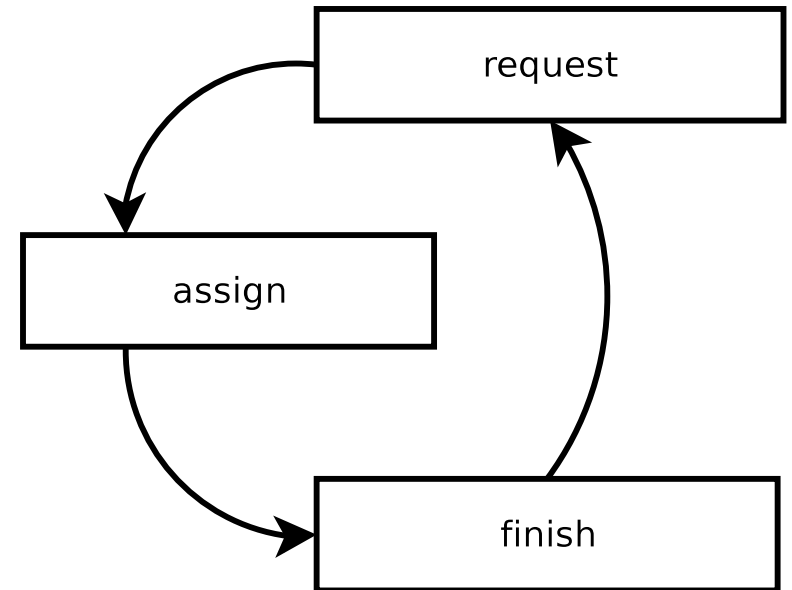
Modal Operators

Modal Fixed-Point Operators

Conclusions

A General Type Scheduler :

Client requests a Resource
Client enqueued
Client assigned a Resource
Client returns the Resource



Polynomial Functors and (Co-)Algebras

Introduction

Preliminaries

● Polynomial Functors and
(Co-)Algebras

● Polynomial Types

● Signatures in CCSL

● The Logic of CCSL

Modal Operators

Modal Fixed-Point Operators

Conclusions

Polynomial Functors T are defined by

$$TX = A \mid X \mid T_1X \times T_2X \mid T_1X + T_2X \mid A \rightarrow T_1X$$

for T_1 and T_2 being polynomial functors,
and A an arbitrary set

Polynomial Functors and (Co-)Algebras

Introduction

Preliminaries

● Polynomial Functors and
(Co-)Algebras

● Polynomial Types

● Signatures in CCSL

● The Logic of CCSL

Modal Operators

Modal Fixed-Point Operators

Conclusions

Polynomial Functors T are defined by

$$TX = A \mid X \mid T_1X \times T_2X \mid T_1X + T_2X \mid A \rightarrow T_1X$$

A T -algebra alg_T is a function

$$alg_T : TX \rightarrow X$$

■ Algebras describe Data-Types

Polynomial Functors and (Co-)Algebras

Introduction

Preliminaries

● Polynomial Functors and (Co-)Algebras

● Polynomial Types

● Signatures in CCSL

● The Logic of CCSL

Modal Operators

Modal Fixed-Point Operators

Conclusions

Polynomial Functors T are defined by

$$TX = A \mid X \mid T_1X \times T_2X \mid T_1X + T_2X \mid A \rightarrow T_1X$$

A T -algebra alg_T is a function

$$alg_T : TX \rightarrow X$$

■ Algebras describe Data-Types

A T -coalgebra clg is a function

$$clg_T : X \rightarrow TX$$

■ Coalgebras determine general type LTS

Polynomial Types

Introduction

Preliminaries

● Polynomial Functors and
(Co-)Algebras

● Polynomial Types

● Signatures in CCSL

● The Logic of CCSL

Modal Operators

Modal Fixed-Point Operators

Conclusions

Polynomial Functors :

$$TX = A \mid X \mid T_1X \times T_2X \mid T_1X + T_2X \mid A \rightarrow T_1X$$

Types $\Xi \vdash \tau$ are defined as

■ a type variable α from Ξ

Polynomial Types

Introduction

Preliminaries

● Polynomial Functors and
(Co-)Algebras

● Polynomial Types

● Signatures in CCSL

● The Logic of CCSL

Modal Operators

Modal Fixed-Point Operators

Conclusions

Polynomial Functors :

$$TX = A \mid X \mid T_1X \times T_2X \mid T_1X + T_2X \mid A \rightarrow T_1X$$

Types $\Xi \vdash \tau$ are defined as

- a type variable α from Ξ
- the special type \mathbf{Prop} of propositions

Polynomial Types

Introduction

Preliminaries

● Polynomial Functors and
(Co-)Algebras

● Polynomial Types

● Signatures in CCSL

● The Logic of CCSL

Modal Operators

Modal Fixed-Point Operators

Conclusions

Polynomial Functors :

$$TX = A \mid X \mid T_1X \times T_2X \mid T_1X + T_2X \mid A \rightarrow T_1X$$

Types $\Xi \vdash \tau$ are defined as

- a type variable α from Ξ
- the special type **Prop**
- the unit type 1 , and the empty type 0

Polynomial Types

Introduction

Preliminaries

● Polynomial Functors and (Co-)Algebras

● Polynomial Types

● Signatures in CCSL

● The Logic of CCSL

Modal Operators

Modal Fixed-Point Operators

Conclusions

Polynomial Functors :

$$TX = A \mid X \mid T_1X \times T_2X \mid T_1X + T_2X \mid A \rightarrow T_1X$$

Types $\Xi \vdash \tau$ are defined as

- a type variable α from Ξ
- the special type **Prop**
- the unit type 1 , and the empty type 0
- the special type **Self** for the state space

Polynomial Types

Introduction

Preliminaries

● Polynomial Functors and
(Co-)Algebras

● Polynomial Types

● Signatures in CCSL

● The Logic of CCSL

Modal Operators

Modal Fixed-Point Operators

Conclusions

Polynomial Functors :

$$TX = A \mid X \mid T_1X \times T_2X \mid T_1X + T_2X \mid A \rightarrow T_1X$$

Types $\Xi \vdash \tau$ are defined as

- a type variable α from Ξ
- the special type **Prop**
- the unit type 1 , and the empty type 0
- the special type **Self**
- the product $\tau \times \sigma$ of types τ and σ

Polynomial Types

Introduction

Preliminaries

● Polynomial Functors and
(Co-)Algebras

● Polynomial Types

● Signatures in CCSL

● The Logic of CCSL

Modal Operators

Modal Fixed-Point Operators

Conclusions

Polynomial Functors :

$$TX = A \mid X \mid T_1X \times T_2X \mid T_1X + T_2X \mid A \rightarrow T_1X$$

Types $\Xi \vdash \tau$ are defined as

- a type variable α from Ξ
- the special type **Prop**
- the unit type 1 , and the empty type 0
- the special type **Self**
- the product $\tau \times \sigma$
- the coproduct $\tau + \sigma$ of types τ and σ

Polynomial Types

Introduction

Preliminaries

● Polynomial Functors and (Co-)Algebras

● Polynomial Types

● Signatures in CCSL

● The Logic of CCSL

Modal Operators

Modal Fixed-Point Operators

Conclusions

Polynomial Functors :

$$TX = A \mid X \mid T_1X \times T_2X \mid T_1X + T_2X \mid A \rightarrow T_1X$$

Types $\Xi \vdash \tau$ are defined as

- a type variable α from Ξ
- the special type **Prop**
- the unit type 1 , and the empty type 0
- the special type **Self**
- the product $\tau \times \sigma$
- the coproduct $\tau + \sigma$
- the exponent $\tau \rightarrow \sigma$ for types τ and σ whereby τ does not contain any occurrence of **Self**

Polynomial Types

Introduction

Preliminaries

● Polynomial Functors and
(Co-)Algebras

● Polynomial Types

● Signatures in CCSL

● The Logic of CCSL

Modal Operators

Modal Fixed-Point Operators

Conclusions

Polynomial Functors :

$$TX = A \mid X \mid T_1X \times T_2X \mid T_1X + T_2X \mid A \rightarrow T_1X$$

Types $\Xi \vdash \tau$ are defined as

- a type variable α from Ξ
- the special type **Prop**
- the unit type 1 , and the empty type 0
- the special type **Self**
- the product $\tau \times \sigma$
- the coproduct $\tau + \sigma$
- the exponent $\tau \rightarrow \sigma$
- $C[\tau_1, \dots, \tau_k]$ for a type constructor of arity k

Signatures in CCSL

Introduction

Preliminaries

- Polynomial Functors and (Co-)Algebras
- Polynomial Types
- Signatures in CCSL
- The Logic of CCSL

Modal Operators

Modal Fixed-Point Operators

Conclusions

A Signature over a set \mathcal{C} of Type Constructors and a set \mathcal{P} of Type Parameters is a tuple $\Sigma = \langle \Sigma_M, \Sigma_C \rangle$

Example : $\mathcal{C} = \{Lift\}$ and $\mathcal{P} = \{Client, Resource\}$

Signatures in CCSL

Introduction

Preliminaries

- Polynomial Functors and (Co-)Algebras
- Polynomial Types
- Signatures in CCSL
- The Logic of CCSL

Modal Operators

Modal Fixed-Point Operators

Conclusions

A Signature over a set \mathcal{C} of Type Constructors and a set \mathcal{P} of Type Parameters is a tuple $\Sigma = \langle \Sigma_M, \Sigma_C \rangle$

Example : $\mathcal{C} = \{Lift\}$ and $\mathcal{P} = \{Client, Resource\}$

$$\Sigma_M \quad \{m : \tau\}$$

Signatures in CCSL

Introduction

Preliminaries

- Polynomial Functors and (Co-)Algebras
- Polynomial Types
- Signatures in CCSL
- The Logic of CCSL

Modal Operators

Modal Fixed-Point Operators

Conclusions

A Signature over a set \mathcal{C} of Type Constructors and a set \mathcal{P} of Type Parameters is a tuple $\Sigma = \langle \Sigma_M, \Sigma_C \rangle$

Example : $\mathcal{C} = \{Lift\}$ and $\mathcal{P} = \{Client, Resource\}$

$\Sigma_M \quad \{m : \tau\}$

request : *Client* \rightarrow **Self**

assign : *Client* \times *Resource* \rightarrow **Self**

finished : *Client* \times *Resource* \rightarrow **Self**

get_resource : *Lift*[*Resource*]

Signatures in CCSL

Introduction

Preliminaries

- Polynomial Functors and (Co-)Algebras
- Polynomial Types
- Signatures in CCSL
- The Logic of CCSL

Modal Operators

Modal Fixed-Point Operators

Conclusions

A Signature over a set \mathcal{C} of Type Constructors and a set \mathcal{P} of Type Parameters is a tuple $\Sigma = \langle \Sigma_M, \Sigma_C \rangle$

Example : $\mathcal{C} = \{Lift\}$ and $\mathcal{P} = \{Client, Resource\}$

$$\Sigma_M \quad \{m : \tau\}$$
$$\Sigma_C \quad \{c : \sigma\}$$

request : *Client* → **Self**
assign : *Client* × *Resource* → **Self**
finished : *Client* × *Resource* → **Self**
get_resource : *Lift*[*Resource*]

Signatures in CCSL

Introduction

Preliminaries

- Polynomial Functors and (Co-)Algebras
- Polynomial Types
- Signatures in CCSL
- The Logic of CCSL

Modal Operators

Modal Fixed-Point Operators

Conclusions

A Signature over a set \mathcal{C} of Type Constructors and a set \mathcal{P} of Type Parameters is a tuple $\Sigma = \langle \Sigma_M, \Sigma_C \rangle$

Example : $\mathcal{C} = \{Lift\}$ and $\mathcal{P} = \{Client, Resource\}$

Σ_M	$\{m : \tau\}$	$request : Client \rightarrow \mathbf{Self}$
		$assign : Client \times Resource \rightarrow \mathbf{Self}$
		$finished : Client \times Resource \rightarrow \mathbf{Self}$
		$get_resource : Lift[Resource]$
Σ_C	$\{c : \sigma\}$	$new_scheduler : 1$

Signatures in CCSL

Introduction

Preliminaries

- Polynomial Functors and (Co-)Algebras
- Polynomial Types
- Signatures in CCSL
- The Logic of CCSL

Modal Operators

Modal Fixed-Point Operators

Conclusions

A Signature over a set \mathcal{C} of Type Constructors and a set \mathcal{P} of Type Parameters is a tuple $\Sigma = \langle \Sigma_M, \Sigma_C \rangle$

Example : $\mathcal{C} = \{Lift\}$ and $\mathcal{P} = \{Client, Resource\}$

$$\Sigma_M \quad \{m : \tau\}$$
$$\Sigma_C \quad \{c : \sigma\}$$

$request : Client \rightarrow \mathbf{Self}$
 $assign : Client \times Resource \rightarrow \mathbf{Self}$
 $finished : Client \times Resource \rightarrow \mathbf{Self}$
 $get_resource : Lift[Resource]$
 $new_scheduler : 1$

Semantics of Signatures:

- a Model of a Signature is a triple $\langle X, c, a \rangle$

The Logic of CCSL

Introduction

Preliminaries

- Polynomial Functors and (Co-)Algebras
- Polynomial Types
- Signatures in CCSL
- The Logic of CCSL

Modal Operators

Modal Fixed-Point Operators

Conclusions

Terms $\Xi | \Gamma \vdash t : \tau$ are defined as

- Ground Terms : constant symbols $c : \tau$, method symbols $m : \sigma$, constructor symbols $c : \rho$

The Logic of CCSL

Introduction

Preliminaries

- Polynomial Functors and (Co-)Algebras
- Polynomial Types
- Signatures in CCSL
- The Logic of CCSL

Modal Operators

Modal Fixed-Point Operators

Conclusions

Terms $\Xi|\Gamma \vdash t : \tau$ are defined as

- Ground Terms : constant symbols $c : \tau$, method symbols $m : \sigma$, constructor symbols $c : \rho$
- Variable Terms : typed variables $x : \sigma$

The Logic of CCSL

Introduction

Preliminaries

- Polynomial Functors and (Co-)Algebras
- Polynomial Types
- Signatures in CCSL
- The Logic of CCSL

Modal Operators

Modal Fixed-Point Operators

Conclusions

Terms $\Xi|\Gamma \vdash t : \tau$ are defined as

- Ground Terms : constant symbols $c : \tau$, method symbols $m : \sigma$, constructor symbols $c : \rho$
- Variable Terms : typed variables $x : \sigma$
- Equality Terms : behavioural equality $t_1 \sim t_2$, and equality $t_1 = t_2$

The Logic of CCSL

Introduction

Preliminaries

- Polynomial Functors and (Co-)Algebras
- Polynomial Types
- Signatures in CCSL
- The Logic of CCSL

Modal Operators

Modal Fixed-Point Operators

Conclusions

Terms $\Xi | \Gamma \vdash t : \tau$ are defined as

- Ground Terms : constant symbols $c : \tau$, method symbols $m : \sigma$, constructor symbols $c : \rho$
- Variable Terms : typed variables $x : \sigma$
- Equality Terms : behavioural equality $t_1 \sim t_2$, and equality $t_1 = t_2$
- Constructions over :
 - ◆ Abstraction $\lambda(x : \sigma).t_1$
 - ◆ Application $t_1 t_2 : \tau$
 - ◆ binary Product $(t_1, t_2) : \sigma_1 \times \tau_1$ with Proj. $\pi_i(t_1, t_2) : \sigma_i$
 - ◆ Injections $\kappa_1 t_1 : \sigma + \tau$ and $\kappa_2 t_2 : \sigma + \tau$
 - ◆ Boolean Connectives $\neg, \vee, \wedge, \Rightarrow, \iff$
 - ◆ Quantifiers \forall, \exists
 - ◆ infinitary Modal Operators ■, ◆

The Logic of CCSL

Introduction

Preliminaries

- Polynomial Functors and (Co-)Algebras
- Polynomial Types
- Signatures in CCSL
- The Logic of CCSL

Modal Operators

Modal Fixed-Point Operators

Conclusions

Terms $\Xi | \Gamma \vdash t : \tau$ are defined as

- Ground Terms : constant symbols $c : \tau$, method symbols $m : \sigma$, constructor symbols $c : \rho$
- Variable Terms : typed variables $x : \sigma$
- Equality Terms : behavioural equality $t_1 \sim t_2$, and equality $t_1 = t_2$
- Constructions over :
 - ◆ Abstraction $\lambda(x : \sigma).t_1$
 - ◆ Application $t_1 t_2 : \tau$
 - ◆ binary Product $(t_1, t_2) : \sigma_1 \times \tau_1$ with Proj. $\pi_i(t_1, t_2) : \sigma_i$
 - ◆ Injections $\kappa_1 t_1 : \sigma + \tau$ and $\kappa_2 t_2 : \sigma + \tau$
 - ◆ Boolean Connectives $\neg, \vee, \wedge, \Rightarrow, \iff$
 - ◆ Quantifiers \forall, \exists
 - ◆ infinitary Modal Operators ■, ◆

Formulae in CCSL are Terms of type **Prop**

Modal Operators - Introduction

Introduction

Preliminaries

Modal Operators

- Modal Operators - Introduction
- How to separate Successor States
- Paths through polynomial Functors
- Successors by Paths
- Modal Operators

Modal Fixed-Point Operators

Conclusions

In 2000, Jan Rothe defined the Modal Operators \blacksquare^M and \blacklozenge^M in his Master Thesis.

- Method Granularity, only
- infinitary, only

Modal Operators - Introduction

Introduction

Preliminaries

Modal Operators

● Modal Operators -
Introduction

- How to separate Successor
States
- Paths through polynomial
Functors
- Successors by Paths
- Modal Operators

Modal Fixed-Point Operators

Conclusions

In 2000, Jan Rothe defined the Modal Operators \blacksquare^M and \blacklozenge^M in his Master Thesis.

- Method Granularity, only
- infinitary, only

I define single-step Modal Operators \square^P and \diamond^P .

- Path Granularity
- single-step

Modal Operators - Introduction

Introduction

Preliminaries

Modal Operators

● Modal Operators -
Introduction

● How to separate Successor
States

● Paths through polynomial
Functors

● Successors by Paths

● Modal Operators

Modal Fixed-Point Operators

Conclusions

In 2000, Jan Rothe defined the Modal Operators \blacksquare^M and \blacklozenge^M in his Master Thesis.

- Method Granularity, only
- infinitary, only

I define single-step Modal Operators \square^P and \diamond^P .

- Path Granularity
- single-step

Outline of this Chapter :

- Paths through Polynomial Functors
- Successor Function using Paths
- Definition of the Modal Operators

How to separate Successor States

Introduction

Preliminaries

Modal Operators

- Modal Operators - Introduction
- How to separate Successor States
- Paths through polynomial Functors
- Successors by Paths
- Modal Operators

Modal Fixed-Point Operators

Conclusions

Recall from the Scheduler Example :

$$\Sigma_M = \left\{ \begin{array}{l} request : Client \rightarrow \mathbf{Self} \\ assign : Client \times Resource \rightarrow \mathbf{Self} \\ finished : Client \times Resource \rightarrow \mathbf{Self} \\ get_resource : Lift[Resource] \end{array} \right\}$$

How to separate Successor States

Introduction

Preliminaries

Modal Operators

● Modal Operators -

Introduction

● How to separate Successor States

● Paths through polynomial

Functors

● Successors by Paths

● Modal Operators

Modal Fixed-Point Operators

Conclusions

Recall from the Scheduler Example :

$$\Sigma_M = \left\{ \begin{array}{l} request : Client \rightarrow \mathbf{Self} \\ assign : Client \times Resource \rightarrow \mathbf{Self} \\ finished : Client \times Resource \rightarrow \mathbf{Self} \\ get_resource : Lift[Resource] \end{array} \right\}$$

- Successor States by Methods [Method Granularity]
- Successor States by Paths [Path Granularity]

Paths through polynomial Functors

Introduction

Preliminaries

Modal Operators

- Modal Operators - Introduction
- How to separate Successor States
- Paths through polynomial Functors
- Successors by Paths
- Modal Operators

Modal Fixed-Point Operators

Conclusions

Polynomial Functors :

$$TX = A \mid X \mid T_1X \times T_2X \mid T_1X + T_2X \mid A \rightarrow T_1X$$

Paths through polynomial Functors :

■ $P(A) = \{*\}$

Paths through polynomial Functors

Introduction

Preliminaries

Modal Operators

- Modal Operators - Introduction
- How to separate Successor States
- Paths through polynomial Functors
- Successors by Paths
- Modal Operators

Modal Fixed-Point Operators

Conclusions

Polynomial Functors :

$$TX = A \mid X \mid T_1X \times T_2X \mid T_1X + T_2X \mid A \rightarrow T_1X$$

Paths through polynomial Functors :

- $P(A) = \{*\}$
- $P(X) = \{*\}$

Paths through polynomial Functors

Introduction

Preliminaries

Modal Operators

- Modal Operators - Introduction
- How to separate Successor States
- Paths through polynomial Functors
- Successors by Paths
- Modal Operators

Modal Fixed-Point Operators

Conclusions

Polynomial Functors :

$$TX = A \mid X \mid T_1X \times T_2X \mid T_1X + T_2X \mid A \rightarrow T_1X$$

Paths through polynomial Functors :

- $P(A) = \{*\}$
- $P(X) = \{*\}$
- $P(T_1 \times T_2) = \hat{\kappa}_1 P(T_1) \cup \hat{\kappa}_2 P(T_2)$

Paths through polynomial Functors

Introduction

Preliminaries

Modal Operators

- Modal Operators - Introduction
- How to separate Successor States
- Paths through polynomial Functors
- Successors by Paths
- Modal Operators

Modal Fixed-Point Operators

Conclusions

Polynomial Functors :

$$TX = A \mid X \mid T_1X \times T_2X \mid T_1X + T_2X \mid A \rightarrow T_1X$$

Paths through polynomial Functors :

- $P(A) = \{*\}$
- $P(X) = \{*\}$
- $P(T_1 \times T_2) = \hat{\kappa}_1 P(T_1) \cup \hat{\kappa}_2 P(T_2)$
- $P(T_1 + T_2) = \hat{\kappa}_1 P(T_1) \cup \hat{\kappa}_2 P(T_2)$

Paths through polynomial Functors

Introduction

Preliminaries

Modal Operators

- Modal Operators - Introduction
- How to separate Successor States
- Paths through polynomial Functors
- Successors by Paths
- Modal Operators

Modal Fixed-Point Operators

Conclusions

Polynomial Functors :

$$TX = A \mid X \mid T_1X \times T_2X \mid T_1X + T_2X \mid A \rightarrow T_1X$$

Paths through polynomial Functors :

- $P(A) = \{*\}$
- $P(X) = \{*\}$
- $P(T_1 \times T_2) = \hat{\kappa}_1 P(T_1) \cup \hat{\kappa}_2 P(T_2)$
- $P(T_1 + T_2) = \hat{\kappa}_1 P(T_1) \cup \hat{\kappa}_2 P(T_2)$
- $P(A \rightarrow T_1) = A \times P(T_1)$

Successors by Paths

Introduction

Preliminaries

Modal Operators

- Modal Operators - Introduction
- How to separate Successor States
- Paths through polynomial Functors
- Successors by Paths
- Modal Operators

Modal Fixed-Point Operators

Conclusions

$$\text{succ}_T : TX \times P_T \rightarrow X + \{*\}$$

- For $x \in X$ we get $\text{succ}_T(x, *) = \kappa_1 x$

succ_T naturally extends to sets of Paths :

$$\hat{\text{succ}}_T(x, P) = \{x' \mid \exists p \in P. \kappa_1 x' = \text{succ}_T(x, p)\}$$

Successors by Paths

Introduction

Preliminaries

Modal Operators

- Modal Operators - Introduction
- How to separate Successor States
- Paths through polynomial Functors
- Successors by Paths
- Modal Operators

Modal Fixed-Point Operators

Conclusions

$$succ_T : TX \times P_T \rightarrow X + \{*\}$$

- For $x \in X$ we get $succ_T(x, *) = \kappa_1 x$
- For $a \in A$ we get $succ_T(a, *) = \kappa_2 a$

$succ_T$ naturally extends to sets of Paths :

$$\hat{succ}_T(x, P) = \{x' \mid \exists p \in P. \kappa_1 x' = succ_T(x, p)\}$$

Successors by Paths

Introduction

Preliminaries

Modal Operators

- Modal Operators - Introduction
- How to separate Successor States
- Paths through polynomial Functors
- Successors by Paths
- Modal Operators

Modal Fixed-Point Operators

Conclusions

$$\text{succ}_T : TX \times P_T \rightarrow X + \{*\}$$

- For $x \in X$ we get $\text{succ}_T(x, *) = \kappa_1 x$
- For $a \in A$ we get $\text{succ}_T(a, *) = \kappa_2 *$
- For $x \in (T_1 X \times T_2 X)$
 $\text{succ}_T(x, \kappa_1 p) = \text{succ}_t(\pi_1 x, p)$
 $\text{succ}_T(x, \kappa_2 p) = \text{succ}_t(\pi_2 x, p)$

succ_T naturally extends to sets of Paths :

$$\hat{\text{succ}}_T(x, P) = \{x' \mid \exists p \in P. \kappa_1 x' = \text{succ}_T(x, p)\}$$

Successors by Paths

Introduction

Preliminaries

Modal Operators

- Modal Operators - Introduction
- How to separate Successor States
- Paths through polynomial Functors
- Successors by Paths
- Modal Operators

Modal Fixed-Point Operators

Conclusions

$$succ_T : TX \times P_T \rightarrow X + \{*\}$$

- For $x \in X$ we get $succ_T(x, *) = \kappa_1 x$
- For $a \in A$ we get $succ_T(a, *) = \kappa_2 *$
- For $x \in (T_1 X \times T_2 X)$
 - $succ_T(x, \kappa_1 p) = succ_t(\pi_1 x, p)$
 - $succ_T(x, \kappa_2 p) = succ_t(\pi_2 x, p)$
- For $\kappa_i x \in T_1 X + T_2 X$
 - $succ_T(\kappa_1 x, \kappa_1 p) = succ_t(x, p)$
 - $succ_T(\kappa_2 x, \kappa_2 p) = succ_t(x, p)$
 - $succ_T(\kappa_1 x, \kappa_2 p) = \kappa_2 *$
 - $succ_T(\kappa_2 x, \kappa_1 p) = \kappa_2 *$

$succ_T$ naturally extends to sets of Paths :

$$\hat{succ}_T(x, P) = \{x' \mid \exists p \in P. \kappa_1 x' = succ_T(x, p)\}$$

Successors by Paths

Introduction

Preliminaries

Modal Operators

- Modal Operators - Introduction
- How to separate Successor States
- Paths through polynomial Functors

● Successors by Paths

● Modal Operators

Modal Fixed-Point Operators

Conclusions

$$succ_T : TX \times P_T \rightarrow X + \{*\}$$

- For $x \in X$ we get $succ_T(x, *) = \kappa_1 x$
- For $a \in A$ we get $succ_T(a, *) = \kappa_2 *$
- For $x \in (T_1 X \times T_2 X)$
 - $succ_T(x, \kappa_1 p) = succ_t(\pi_1 x, p)$
 - $succ_T(x, \kappa_2 p) = succ_t(\pi_2 x, p)$
- For $\kappa_i x \in T_1 X + T_2 X$
 - $succ_T(\kappa_1 x, \kappa_1 p) = succ_t(x, p)$
 - $succ_T(\kappa_2 x, \kappa_2 p) = succ_t(x, p)$
 - $succ_T(\kappa_1 x, \kappa_2 p) = \kappa_2 *$
 - $succ_T(\kappa_2 x, \kappa_1 p) = \kappa_2 *$
- For $x : A \rightarrow T_1 X$ we get $succ_T(x, (a, p)) = succ_T(x(a), p)$

$succ_T$ naturally extends to sets of Paths :

$$\hat{succ}_T(x, P) = \{x' \mid \exists p \in P. \kappa_1 x' = succ_T(x, p)\}$$

Modal Operators

Introduction

Preliminaries

Modal Operators

- Modal Operators - Introduction
- How to separate Successor States
- Paths through polynomial Functors
- Successors by Paths
- Modal Operators

Modal Fixed-Point Operators

Conclusions

Syntax of the single-step Modal Operators :

■ $\Box^P F : \mathbf{Self} \rightarrow \mathbf{Prop}$

■ $\Diamond^P F : \mathbf{Self} \rightarrow \mathbf{Prop}$

for $F : \mathbf{Self} \rightarrow \mathbf{Prop}$

Modal Operators

Introduction

Preliminaries

Modal Operators

- Modal Operators - Introduction
- How to separate Successor States
- Paths through polynomial Functors
- Successors by Paths
- Modal Operators

Modal Fixed-Point Operators

Conclusions

Syntax of the single-step Modal Operators :

■ $\Box^P F : \mathbf{Self} \rightarrow \mathbf{Prop}$

■ $\Diamond^P F : \mathbf{Self} \rightarrow \mathbf{Prop}$

for $F : \mathbf{Self} \rightarrow \mathbf{Prop}$

Semantics of the single-step Modal Operators :

$$\llbracket \Box^P F \rrbracket = \lambda(\bar{x} : \bar{\sigma}). \lambda(x : X). \forall x' \in X. (x' \in \hat{succ}_T(cx, P) \Rightarrow \llbracket F \rrbracket(x', \bar{x}))$$

Modal Operators

Introduction

Preliminaries

Modal Operators

- Modal Operators - Introduction
- How to separate Successor States
- Paths through polynomial Functors
- Successors by Paths
- Modal Operators

Modal Fixed-Point Operators

Conclusions

Syntax of the single-step Modal Operators :

■ $\Box^P F : \mathbf{Self} \rightarrow \mathbf{Prop}$

■ $\Diamond^P F : \mathbf{Self} \rightarrow \mathbf{Prop}$

for $F : \mathbf{Self} \rightarrow \mathbf{Prop}$

Semantics of the single-step Modal Operators :

$$\llbracket \Box^P F \rrbracket = \lambda(\bar{x} : \bar{\sigma}). \lambda(x : X). \forall x' \in X. (x' \in \mathit{succ}_T(cx, P) \Rightarrow \llbracket F \rrbracket(x', \bar{x}))$$

\Diamond^P is defined as an Abbreviation :

$$\Diamond^P \lambda(x : \mathbf{Self}). \phi \stackrel{def}{=} \neg \Box^P \lambda(x : \mathbf{Self}). \neg \phi$$

Fixed-Points of Predicate Transformers

Introduction

Preliminaries

Modal Operators

Modal Fixed-Point Operators

- Fixed-Points of Predicate Transformers
- Modal Fixed-Point Operators
- Normalforms
- The syntactic Monotonicity Criterion
- The syntactic Monotonicity Criterion 1,2
- The syntactic Monotonicity Criterion 2,3
- The syntactic Monotonicity Criterion 4
- The syntactic Monotonicity Criterion 5
- Example - Some Temporal Connectives
- Example - Scheduler
- Example - Scheduler

Conclusions

A Predicate-Transformer F over \mathbf{Self} is a Function

$$F : (\mathbf{Self} \rightarrow \mathbf{Prop}) \rightarrow (\mathbf{Self} \rightarrow \mathbf{Prop})$$

Fixed-Points of Predicate Transformers

Introduction

Preliminaries

Modal Operators

Modal Fixed-Point Operators

● Fixed-Points of Predicate Transformers

● Modal Fixed-Point Operators

● Normalforms

● The syntactic Monotonicity Criterion

● The syntactic Monotonicity Criterion 1,2

● The syntactic Monotonicity Criterion 2,3

● The syntactic Monotonicity Criterion 4

● The syntactic Monotonicity Criterion 5

● Example - Some Temporal Connectives

● Example - Scheduler

● Example - Scheduler

Conclusions

A Predicate-Transformer F over \mathbf{Self} is a Function

$$F : (\mathbf{Self} \rightarrow \mathbf{Prop}) \rightarrow (\mathbf{Self} \rightarrow \mathbf{Prop})$$

A Fixed-Point $Z : (\mathbf{Self} \rightarrow \mathbf{Prop})$ of F is determined by

$$F(Z) = Z$$

Fixed-Points of Predicate Transformers

Introduction

Preliminaries

Modal Operators

Modal Fixed-Point Operators

● Fixed-Points of Predicate Transformers

● Modal Fixed-Point Operators

● Normalforms

● The syntactic Monotonicity Criterion

● The syntactic Monotonicity Criterion 1,2

● The syntactic Monotonicity Criterion 2,3

● The syntactic Monotonicity Criterion 4

● The syntactic Monotonicity Criterion 5

● Example - Some Temporal Connectives

● Example - Scheduler

● Example - Scheduler

Conclusions

A Predicate-Transformer F over \mathbf{Self} is a Function

$$F : (\mathbf{Self} \rightarrow \mathbf{Prop}) \rightarrow (\mathbf{Self} \rightarrow \mathbf{Prop})$$

A Fixed-Point $Z : (\mathbf{Self} \rightarrow \mathbf{Prop})$ of F is determined by

$$F(Z) = Z$$

We are interested in the least and greatest Fixed-Points, only.

- may not exist
- Existence is guaranteed by Knaster-Tarski Theorem
- if F is monotonic

Fixed-Points of Predicate Transformers

Introduction

Preliminaries

Modal Operators

Modal Fixed-Point Operators

● Fixed-Points of Predicate Transformers

● Modal Fixed-Point Operators

● Normalforms

● The syntactic Monotonicity Criterion

● The syntactic Monotonicity Criterion 1,2

● The syntactic Monotonicity Criterion 2,3

● The syntactic Monotonicity Criterion 4

● The syntactic Monotonicity Criterion 5

● Example - Some Temporal Connectives

● Example - Scheduler

● Example - Scheduler

Conclusions

A Predicate-Transformer F over \mathbf{Self} is a Function

$$F : (\mathbf{Self} \rightarrow \mathbf{Prop}) \rightarrow (\mathbf{Self} \rightarrow \mathbf{Prop})$$

A Fixed-Point $Z : (\mathbf{Self} \rightarrow \mathbf{Prop})$ of F is determined by

$$F(Z) = Z$$

We are interested in the least and greatest Fixed-Points, only.

- may not exist
- Existence is guaranteed by Knaster-Tarski Theorem
- if F is monotonic

F is monotonic iff $Z_1 \subseteq Z_2 \Rightarrow FZ_1 \subseteq FZ_2$

$Z_1 \subseteq Z_2$ iff $\forall(x : \mathbf{Self}). Z_1(x) \Rightarrow Z_2(x)$

Modal Fixed-Point Operators

Introduction

Preliminaries

Modal Operators

Modal Fixed-Point Operators

● Fixed-Points of Predicate

Transformers

● Modal Fixed-Point Operators

● Normalforms

● The syntactic Monotonicity
Criterion

● The syntactic Monotonicity
Criterion 1,2

● The syntactic Monotonicity
Criterion 2,3

● The syntactic Monotonicity
Criterion 4

● The syntactic Monotonicity
Criterion 5

● Example - Some Temporal
Connectives

● Example - Scheduler

● Example - Scheduler

Conclusions

Syntax of the Modal Fixed-Point Operators :

■ $\mu(\lambda(Z : \mathbf{Self} \rightarrow \mathbf{Prop}).F) : \mathbf{Self} \rightarrow \mathbf{Prop}$

■ $\nu(\lambda(Z : \mathbf{Self} \rightarrow \mathbf{Prop}).F) : \mathbf{Self} \rightarrow \mathbf{Prop}$

$(\lambda(Z : \mathbf{Self} \rightarrow \mathbf{Prop}).F)$ has to be monotonic)

Modal Fixed-Point Operators

Introduction

Preliminaries

Modal Operators

Modal Fixed-Point Operators

● Fixed-Points of Predicate

Transformers

● Modal Fixed-Point Operators

● Normalforms

● The syntactic Monotonicity
Criterion

● The syntactic Monotonicity
Criterion 1,2

● The syntactic Monotonicity
Criterion 2,3

● The syntactic Monotonicity
Criterion 4

● The syntactic Monotonicity
Criterion 5

● Example - Some Temporal
Connectives

● Example - Scheduler

● Example - Scheduler

Conclusions

Syntax of the Modal Fixed-Point Operators :

■ $\mu(\lambda(Z : \mathbf{Self} \rightarrow \mathbf{Prop}).F) : \mathbf{Self} \rightarrow \mathbf{Prop}$

■ $\nu(\lambda(Z : \mathbf{Self} \rightarrow \mathbf{Prop}).F) : \mathbf{Self} \rightarrow \mathbf{Prop}$

$(\lambda(Z : \mathbf{Self} \rightarrow \mathbf{Prop}).F)$ has to be monotonic)

Semantics of the Modal Fixed-Point Operators :

$$\begin{aligned} \llbracket \mu(\lambda(Z : \mathbf{Self} \rightarrow \mathbf{Prop}).F) \rrbracket_{U_1, \dots, U_n} = \\ \lambda \bar{x} : \bar{\sigma}. \text{lfp}(\lambda Z : X \rightarrow 2. \llbracket F \rrbracket_{U_1, \dots, U_n}(\bar{x}.Z)) \end{aligned}$$

ν is defined as an Abbreviation as follows ($F = \lambda(x : \mathbf{Self}).\phi$):

$$\begin{aligned} \nu(\lambda(Z : \mathbf{Self} \rightarrow \mathbf{Prop}).\lambda(x : \mathbf{Self}).\phi) = \\ \mu(Z : \mathbf{Self} \rightarrow \mathbf{Prop}).\lambda(x : \mathbf{Self}).\neg\phi[Z/\neg Z] \end{aligned}$$

Normalforms

Introduction

Preliminaries

Modal Operators

Modal Fixed-Point Operators

- Fixed-Points of Predicate Transformers
- Modal Fixed-Point Operators

● Normalforms

- The syntactic Monotonicity Criterion
- The syntactic Monotonicity Criterion 1,2
- The syntactic Monotonicity Criterion 2,3
- The syntactic Monotonicity Criterion 4
- The syntactic Monotonicity Criterion 5
- Example - Some Temporal Connectives
- Example - Scheduler
- Example - Scheduler

Conclusions

Beta - Normalform :

■ $(\lambda(x : \sigma).t')s \longrightarrow_{\beta} t'[s/x]$ for $s : \sigma$ being a term

Normalforms

Introduction

Preliminaries

Modal Operators

Modal Fixed-Point Operators

● Fixed-Points of Predicate

Transformers

● Modal Fixed-Point Operators

● Normalforms

● The syntactic Monotonicity
Criterion

● The syntactic Monotonicity
Criterion 1,2

● The syntactic Monotonicity
Criterion 2,3

● The syntactic Monotonicity
Criterion 4

● The syntactic Monotonicity
Criterion 5

● Example - Some Temporal
Connectives

● Example - Scheduler

● Example - Scheduler

Conclusions

Beta - Normalform :

$$\blacksquare (\lambda(x : \sigma).t')s \longrightarrow_{\beta} t'[s/x]$$

Negation - Normalform :

$$\blacksquare \neg\neg\phi \longrightarrow_{\neg} \phi$$

$$\blacksquare \neg\top \longrightarrow_{\neg} \perp \text{ and } \neg\perp \longrightarrow_{\neg} \top$$

$$\blacksquare \neg(\phi \vee \psi) \longrightarrow_{\neg} \neg\phi \wedge \neg\psi \text{ and } \neg(\phi \wedge \psi) \longrightarrow_{\neg} \neg\phi \vee \neg\psi$$

$$\blacksquare \neg\forall s : \sigma.\psi \longrightarrow_{\neg} \exists s : \sigma.\neg\psi \text{ and } \neg\exists s : \sigma.\psi \longrightarrow_{\neg} \forall s : \sigma.\neg\psi$$

$$\blacksquare \neg\Box^P \lambda(x : \mathbf{Self}).\phi \longrightarrow_{\neg} \Diamond^P \lambda(x : \mathbf{Self}).\neg\phi \text{ and } \neg\Diamond^P \lambda(x : \mathbf{Self}).\phi \longrightarrow_{\neg} \Box^P \lambda(x : \mathbf{Self}).\neg\phi$$

$$\blacksquare \neg\mu\lambda(Z : \mathbf{Self} \rightarrow \mathbf{Prop}).\lambda(x : \mathbf{Self}).\phi[Z] \longrightarrow_{\neg} \nu\lambda(Z : \mathbf{Self} \rightarrow \mathbf{Prop}).\lambda(x : \mathbf{Self}).\neg\phi[\neg Z/Z] \text{ and } \neg\nu\lambda(Z : \mathbf{Self} \rightarrow \mathbf{Prop}).\lambda(x : \mathbf{Self}).\phi[Z] \longrightarrow_{\neg} \mu\lambda(Z : \mathbf{Self} \rightarrow \mathbf{Prop}).\lambda(x : \mathbf{Self}).\neg\phi[\neg Z/Z]$$

(all Abbreviations are assumed to be expanded)

The syntactic Monotonicity Criterion

Introduction

Preliminaries

Modal Operators

Modal Fixed-Point Operators

- Fixed-Points of Predicate Transformers
- Modal Fixed-Point Operators
- Normalforms

● The syntactic Monotonicity Criterion

- The syntactic Monotonicity Criterion 1,2
- The syntactic Monotonicity Criterion 2,3
- The syntactic Monotonicity Criterion 4
- The syntactic Monotonicity Criterion 5
- Example - Some Temporal Connectives
- Example - Scheduler
- Example - Scheduler

Conclusions

$F = \lambda(Z : \mathbf{Self} \rightarrow \mathbf{Prop}).\lambda(x : \mathbf{Self}).\phi$ is monotonic if ϕ satisfies the syntactic Monotonicity Criterion, i.e. ϕ is in Negation Normalform and

1. ϕ does not contain Z
2. ϕ is of the form Zx or $\phi_1 \wedge \phi_2$ or $\phi_1 \vee \phi_2$
3. ϕ is of the form $\forall(v : \tau).\phi_1$ or $\exists(v : \tau).\phi_1$
4. ϕ is of the form $(\Box^P(\lambda x.\phi_1))(x)$ or $(\Diamond^P(\lambda x.\phi_1))(x)$
5. ϕ is of the form $(\mu(\lambda Z'.\lambda x.\phi_3))(x)$ or $(\nu(\lambda Z'.\lambda x.\phi_3))(x)$

such that

- ϕ_1 and ϕ_2 satisfy the syntactic Monotonicity Criterion w.r.t. Z
- ϕ_3 satisfies the syntactic Monotonicity Criterion w.r.t. Z and Z'
- P valid paths

The syntactic Monotonicity Criterion 1,2

Introduction

Preliminaries

Modal Operators

Modal Fixed-Point Operators

- Fixed-Points of Predicate Transformers
- Modal Fixed-Point Operators
- Normalforms
- The syntactic Monotonicity Criterion
- **The syntactic Monotonicity Criterion 1,2**
- The syntactic Monotonicity Criterion 2,3
- The syntactic Monotonicity Criterion 4
- The syntactic Monotonicity Criterion 5
- Example - Some Temporal Connectives
- Example - Scheduler
- Example - Scheduler

Conclusions

$F = \lambda(Z : \mathbf{Self} \rightarrow \mathbf{Prop}).\lambda(x : \mathbf{Self}).\phi.$

To show : $Z_1 \subseteq Z_2$ implies $\phi[Z_1/Z][x] \Rightarrow \phi[Z_2/Z][x]$ for all x

1. Z does not occur in ϕ
 - then $(\phi[Z_1/Z])[x] \Rightarrow (\phi[Z_2/Z])[x]$ trivially

The syntactic Monotonicity Criterion 1,2

Introduction

Preliminaries

Modal Operators

Modal Fixed-Point Operators

- Fixed-Points of Predicate Transformers
- Modal Fixed-Point Operators
- Normalforms
- The syntactic Monotonicity Criterion
- **The syntactic Monotonicity Criterion 1,2**
- The syntactic Monotonicity Criterion 2,3
- The syntactic Monotonicity Criterion 4
- The syntactic Monotonicity Criterion 5
- Example - Some Temporal Connectives
- Example - Scheduler
- Example - Scheduler

Conclusions

$$F = \lambda(Z : \mathbf{Self} \rightarrow \mathbf{Prop}).\lambda(x : \mathbf{Self}).\phi.$$

To show : $Z_1 \subseteq Z_2$ implies $\phi[Z_1/Z][x] \Rightarrow \phi[Z_2/Z][x]$ for all x

1. Z does not occur in ϕ
 - then $(\phi[Z_1/Z])[x] \Rightarrow (\phi[Z_2/Z])[x]$ trivially
2. $\phi = Zx$
 - then $Z_1x \Rightarrow Z_2x$ by $Z_1 \subseteq Z_2$

The syntactic Monotonicity Criterion 1,2

Introduction

Preliminaries

Modal Operators

Modal Fixed-Point Operators

- Fixed-Points of Predicate Transformers
- Modal Fixed-Point Operators
- Normalforms
- The syntactic Monotonicity Criterion
- **The syntactic Monotonicity Criterion 1,2**
- The syntactic Monotonicity Criterion 2,3
- The syntactic Monotonicity Criterion 4
- The syntactic Monotonicity Criterion 5
- Example - Some Temporal Connectives
- Example - Scheduler
- Example - Scheduler

Conclusions

$F = \lambda(Z : \mathbf{Self} \rightarrow \mathbf{Prop}).\lambda(x : \mathbf{Self}).\phi.$

To show : $Z_1 \subseteq Z_2$ implies $\phi[Z_1/Z][x] \Rightarrow \phi[Z_2/Z][x]$ for all x

1. Z does not occur in ϕ
 - then $(\phi[Z_1/Z])[x] \Rightarrow (\phi[Z_2/Z])[x]$ trivially
2. $\phi = Zx$
 - then $Z_1x \Rightarrow Z_2x$ by $Z_1 \subseteq Z_2$
3. $\phi = \phi_1 \wedge \phi_2$
 - assume $\phi_i[Z_1/Z] \Rightarrow \phi_i[Z_2/Z]$ for $i \in \{1, 2\}$
 - then

$$\left(\begin{array}{c} \phi_1[Z_1/Z][x] \wedge \\ \phi_2[Z_1/Z][x] \end{array} \right) \Rightarrow \left(\begin{array}{c} \phi_1[Z_2/Z][x] \wedge \\ \phi_2[Z_2/Z][x] \end{array} \right)$$

The syntactic Monotonicity Criterion 2,3

Introduction

Preliminaries

Modal Operators

Modal Fixed-Point Operators

- Fixed-Points of Predicate Transformers
- Modal Fixed-Point Operators
- Normalforms
- The syntactic Monotonicity Criterion
- The syntactic Monotonicity Criterion 1,2
- The syntactic Monotonicity Criterion 2,3
- The syntactic Monotonicity Criterion 4
- The syntactic Monotonicity Criterion 5
- Example - Some Temporal Connectives
- Example - Scheduler
- Example - Scheduler

Conclusions

$F = \lambda(Z : \mathbf{Self} \rightarrow \mathbf{Prop}).\lambda(x : \mathbf{Self}).\phi.$

To show : $Z_1 \subseteq Z_2$ implies $\phi[Z_1/Z][x] \Rightarrow \phi[Z_2/Z][x]$ for all x

1. $\phi = \phi_1 \vee \phi_2$

■ assume $\phi_i[Z_1/Z][x] \Rightarrow \phi_i[Z_2/Z][x]$ for $i \in \{1, 2\}$

■ then

$$\left(\begin{array}{c} \phi_1[Z_1/Z][x] \vee \\ \phi_2[Z_1/Z][x] \end{array} \right) \Rightarrow \left(\begin{array}{c} \phi_1[Z_2/Z][x] \vee \\ \phi_2[Z_2/Z][x] \end{array} \right)$$

The syntactic Monotonicity Criterion 2,3

Introduction

Preliminaries

Modal Operators

Modal Fixed-Point Operators

- Fixed-Points of Predicate Transformers
- Modal Fixed-Point Operators
- Normalforms
- The syntactic Monotonicity Criterion
- The syntactic Monotonicity Criterion 1,2
- The syntactic Monotonicity Criterion 2,3
- The syntactic Monotonicity Criterion 4
- The syntactic Monotonicity Criterion 5
- Example - Some Temporal Connectives
- Example - Scheduler
- Example - Scheduler

Conclusions

$$F = \lambda(Z : \mathbf{Self} \rightarrow \mathbf{Prop}).\lambda(x : \mathbf{Self}).\phi.$$

To show : $Z_1 \subseteq Z_2$ implies $\phi[Z_1/Z][x] \Rightarrow \phi[Z_2/Z][x]$ for all x

1. $\phi = \phi_1 \vee \phi_2$

■ assume $\phi_i[Z_1/Z][x] \Rightarrow \phi_i[Z_2/Z][x]$ for $i \in \{1, 2\}$

■ then

$$\left(\begin{array}{c} \phi_1[Z_1/Z][x] \vee \\ \phi_2[Z_1/Z][x] \end{array} \right) \Rightarrow \left(\begin{array}{c} \phi_1[Z_2/Z][x] \vee \\ \phi_2[Z_2/Z][x] \end{array} \right)$$

2. $\phi = \forall(v : \tau).\phi_1$ or $\phi = \exists(v : \tau).\phi_1$

■ assume $\phi_1[Z_1/Z][x] \Rightarrow \phi_1[Z_2/Z][x]$ for all x

■ then $\forall(v : \tau).\phi_1[Z_1/Z][x] \Rightarrow \forall(v : \tau).\phi_1[Z_2/Z][x]$ for all x

■ then $\exists(v : \tau).\phi_1[Z_1/Z][x] \Rightarrow \exists(v : \tau).\phi_1[Z_2/Z][x]$ for all x

The syntactic Monotonicity Criterion 4

Introduction

Preliminaries

Modal Operators

Modal Fixed-Point Operators

- Fixed-Points of Predicate Transformers
- Modal Fixed-Point Operators
- Normalforms
- The syntactic Monotonicity Criterion
- The syntactic Monotonicity Criterion 1,2
- The syntactic Monotonicity Criterion 2,3
- The syntactic Monotonicity Criterion 4
- The syntactic Monotonicity Criterion 5
- Example - Some Temporal Connectives
- Example - Scheduler
- Example - Scheduler

Conclusions

$$F = \lambda(Z : \mathbf{Self} \rightarrow \mathbf{Prop}).\lambda(x : \mathbf{Self}).\phi.$$

To show : $Z_1 \subseteq Z_2$ implies $\phi[Z_1/Z][x] \Rightarrow \phi[Z_2/Z][x]$ for all x

1. $\phi = \Box^P \lambda(x' : \mathbf{Self}).\phi_1$
 - assume $\phi_1[Z_1/Z][x] \Rightarrow \phi_1[Z_2/Z][x]$ for all x
 - then $\phi_1[Z_1/Z][x] \Rightarrow \phi_1[Z_2/Z][x]$ also for all $x' \in \hat{succ}_T(x, P)$, i.e. successors of x for paths P
 - or equivalently $\forall x' \in \hat{succ}_T(x, P).(\phi_1[Z_1/Z][x'] \Rightarrow \phi_1[Z_2/Z][x'])$
 - implying $\forall x' \in \hat{succ}_T(x, P).\phi_1[Z_1/Z][x'] \Rightarrow \forall x' \in \hat{succ}_T(x, P).\phi_1[Z_2/Z][x']$
 - then by definition of \Box we obtain

$$\Box^P.\lambda(x' : \mathbf{Self}).\phi_1[Z_1/Z] \subseteq \Box^P.\lambda(x' : \mathbf{Self}).\phi_1[Z_2/Z]$$

2. a similar argument for $\phi = \Box^P \lambda(x' : \mathbf{Self}).\phi_1$

The syntactic Monotonicity Criterion 5

Introduction

Preliminaries

Modal Operators

Modal Fixed-Point Operators

- Fixed-Points of Predicate Transformers
- Modal Fixed-Point Operators
- Normalforms
- The syntactic Monotonicity Criterion
- The syntactic Monotonicity Criterion 1,2
- The syntactic Monotonicity Criterion 2,3
- The syntactic Monotonicity Criterion 4
- The syntactic Monotonicity Criterion 5
- Example - Some Temporal Connectives
- Example - Scheduler
- Example - Scheduler

Conclusions

$$F = \lambda(Z : \mathbf{Self} \rightarrow \mathbf{Prop}).\lambda(x : \mathbf{Self}).\phi.$$

To show : $Z_1 \subseteq Z_2$ implies $\phi[Z_1/Z][x] \Rightarrow \phi[Z_2/Z][x]$ for all x

1. $\phi = \mu(\lambda(Z' : \mathbf{Self} \rightarrow \mathbf{Prop}).\lambda(x' : \mathbf{Self}).\phi_1)$
 - assume ϕ monotonic w.r.t. Z'
 - the least Fixed-Point of ϕ for Z' is $\bigwedge Z' \mid Z' = \lambda(x' : \mathbf{Self}).\phi_1$
 - assume $Z_1 \subseteq Z_2$
 - then $\phi_1[Z_1] \Rightarrow \phi_1[Z_2]$ for all Z'
 - then $\bigwedge Z' \mid Z' = \lambda(x : \mathbf{Self}).\phi_1[Z_1/Z] \subseteq \bigwedge Z' \mid Z' = \lambda(x : \mathbf{Self}).\phi_1[Z_2/Z]$
 - $\mu(\lambda Z'.\lambda x'.\phi_1[Z_1/Z]) \subseteq \mu(\lambda Z'.\lambda x'.\phi_1[Z_2/Z])$

Example - Some Temporal Connectives

Introduction

Preliminaries

Modal Operators

Modal Fixed-Point Operators

- Fixed-Points of Predicate Transformers
- Modal Fixed-Point Operators
- Normalforms
- The syntactic Monotonicity Criterion
- The syntactic Monotonicity Criterion 1,2
- The syntactic Monotonicity Criterion 2,3
- The syntactic Monotonicity Criterion 4
- The syntactic Monotonicity Criterion 5
- Example - Some Temporal Connectives
- Example - Scheduler
- Example - Scheduler

Conclusions

Every run does only contain states satisfying ϕ .

$$\blacksquare AG.\phi = \nu(Z : (\mathbf{Self} \rightarrow \mathbf{Prop})). \\ (\lambda(x : \mathbf{Self}).\phi(x) \wedge (\Box Z)(x))$$

Example - Some Temporal Connectives

Introduction

Preliminaries

Modal Operators

Modal Fixed-Point Operators

- Fixed-Points of Predicate Transformers
- Modal Fixed-Point Operators
- Normalforms
- The syntactic Monotonicity Criterion
- The syntactic Monotonicity Criterion 1,2
- The syntactic Monotonicity Criterion 2,3
- The syntactic Monotonicity Criterion 4
- The syntactic Monotonicity Criterion 5
- Example - Some Temporal Connectives
- Example - Scheduler
- Example - Scheduler

Conclusions

Every run does only contain states satisfying ϕ .

$$\blacksquare AG.\phi = \nu(Z : (\mathbf{Self} \rightarrow \mathbf{Prop})). \\ (\lambda(x : \mathbf{Self}).\phi(x) \wedge (\Box Z)(x))$$

There is a Run containing a state satisfying ϕ .

$$\blacksquare AF.\phi = \mu(Z : (\mathbf{Self} \rightarrow \mathbf{Prop})). \\ (\lambda(x : \mathbf{Self}).\phi(x) \vee ((\Diamond \top_{\mathbf{Self}})(x) \wedge \Box Z)(x))$$

Example - Some Temporal Connectives

Introduction

Preliminaries

Modal Operators

Modal Fixed-Point Operators

- Fixed-Points of Predicate Transformers
- Modal Fixed-Point Operators
- Normalforms
- The syntactic Monotonicity Criterion
- The syntactic Monotonicity Criterion 1,2
- The syntactic Monotonicity Criterion 2,3
- The syntactic Monotonicity Criterion 4
- The syntactic Monotonicity Criterion 5
- Example - Some Temporal Connectives
- Example - Scheduler
- Example - Scheduler

Conclusions

Every run does only contain states satisfying ϕ .

$$\blacksquare AG.\phi = \nu(Z : (\mathbf{Self} \rightarrow \mathbf{Prop})). \\ (\lambda(x : \mathbf{Self}).\phi(x) \wedge (\Box Z)(x))$$

There is a Run containing a state satisfying ϕ .

$$\blacksquare AF.\phi = \mu(Z : (\mathbf{Self} \rightarrow \mathbf{Prop})). \\ (\lambda(x : \mathbf{Self}).\phi(x) \vee ((\Diamond \top_{\mathbf{Self}})(x) \wedge \Box Z)(x))$$

Every state in a run satisfies ϕ until a state satisfying ψ is reached

$$\blacksquare \phi UNTIL \psi = \nu(Z : (\mathbf{Self} \rightarrow \mathbf{Prop})). \\ (\lambda(x : \mathbf{Self}).\psi(x) \vee (\phi(x) \wedge (\Box Z)(x)))$$

Example - Scheduler

Introduction

Preliminaries

Modal Operators

Modal Fixed-Point Operators

- Fixed-Points of Predicate Transformers
- Modal Fixed-Point Operators
- Normalforms
- The syntactic Monotonicity Criterion
- The syntactic Monotonicity Criterion 1,2
- The syntactic Monotonicity Criterion 2,3
- The syntactic Monotonicity Criterion 4
- The syntactic Monotonicity Criterion 5
- Example - Some Temporal Connectives
- Example - Scheduler
- Example - Scheduler

Conclusions

Recall Σ_M from the Signature for the Scheduler

	TX	
<i>request</i>	$(C \rightarrow X) \times$	$\hat{\kappa}_1 \cdots$
<i>assign</i>	$((C \times R) \rightarrow X) \times$	$\hat{\kappa}_2 \cdots$
<i>finish</i>	$((C \times R) \rightarrow X) \times$	$\hat{\kappa}_3 \cdots$
<i>get_resource</i>	$(R + 1)$	$\hat{\kappa}_4 \cdots$

The following Property describes that
1. after any client returns a resource

$$1. P_1 = \square^{\hat{\kappa}_3}((C \times R) \times \{*\}.P'_1)$$

Example - Scheduler

Introduction

Preliminaries

Modal Operators

Modal Fixed-Point Operators

- Fixed-Points of Predicate Transformers
- Modal Fixed-Point Operators
- Normalforms
- The syntactic Monotonicity Criterion
- The syntactic Monotonicity Criterion 1,2
- The syntactic Monotonicity Criterion 2,3
- The syntactic Monotonicity Criterion 4
- The syntactic Monotonicity Criterion 5
- Example - Some Temporal Connectives
- Example - Scheduler
- Example - Scheduler

Conclusions

Recall Σ_M from the Signature for the Scheduler

	TX	
<i>request</i>	$(C \rightarrow X) \times$	$\hat{\kappa}_1 \cdots$
<i>assign</i>	$((C \times R) \rightarrow X) \times$	$\hat{\kappa}_2 \cdots$
<i>finish</i>	$((C \times R) \rightarrow X) \times$	$\hat{\kappa}_3 \cdots$
<i>get_resource</i>	$(R + 1)$	$\hat{\kappa}_4 \cdots$

The following Property describes that

1. after any client returns a resource
2. a resource will eventually be available

1. $P_1 = \Box^{\hat{\kappa}_3}((C \times R) \times \{*\}).P'_1$
2. $P'_1 = \mu(\lambda(Z : \mathbf{Self} \rightarrow \mathbf{Prop}).\lambda(x' : \mathbf{Self}).(up?(get_resource(x')) \vee ((\Diamond^{\top_{\mathbf{Self}}})(x') \wedge (\Box Z)(x')))))$

Example - Scheduler

Introduction

Preliminaries

Modal Operators

Modal Fixed-Point Operators

- Fixed-Points of Predicate Transformers
- Modal Fixed-Point Operators
- Normalforms
- The syntactic Monotonicity Criterion
- The syntactic Monotonicity Criterion 1,2
- The syntactic Monotonicity Criterion 2,3
- The syntactic Monotonicity Criterion 4
- The syntactic Monotonicity Criterion 5
- Example - Some Temporal Connectives
- Example - Scheduler
- Example - Scheduler

Conclusions

Recall Σ_M from the Signature for the Scheduler

	TX	
<i>request</i>	$(C \rightarrow X) \times$	$\hat{\kappa}_1 \cdots$
<i>assign</i>	$((C \times R) \rightarrow X) \times$	$\hat{\kappa}_2 \cdots$
<i>finish</i>	$((C \times R) \rightarrow X) \times$	$\hat{\kappa}_3 \cdots$
<i>get_resource</i>	$(R + 1)$	$\hat{\kappa}_4 \cdots$

The following Property describes that
1. after a Resource is assigned to a Client

$$1. P_2 = \mu(Z : \mathbf{Self} \rightarrow \mathbf{Prop}). \diamond^{\hat{\kappa}_2} ((C \times R) \times \{*\}) P_2'$$

Example - Scheduler

Introduction

Preliminaries

Modal Operators

Modal Fixed-Point Operators

- Fixed-Points of Predicate Transformers
- Modal Fixed-Point Operators
- Normalforms
- The syntactic Monotonicity Criterion
- The syntactic Monotonicity Criterion 1,2
- The syntactic Monotonicity Criterion 2,3
- The syntactic Monotonicity Criterion 4
- The syntactic Monotonicity Criterion 5
- Example - Some Temporal Connectives
- Example - Scheduler
- Example - Scheduler

Conclusions

Recall Σ_M from the Signature for the Scheduler

	TX	
<i>request</i>	$(C \rightarrow X) \times$	$\hat{\kappa}_1 \cdots$
<i>assign</i>	$((C \times R) \rightarrow X) \times$	$\hat{\kappa}_2 \cdots$
<i>finish</i>	$((C \times R) \rightarrow X) \times$	$\hat{\kappa}_3 \cdots$
<i>get_resource</i>	$(R + 1)$	$\hat{\kappa}_4 \cdots$

The following Property describes that

1. after a Resource is assigned to a Client
2. and after any Client returns a Resource

1. $P_2 = \mu(Z : \mathbf{Self} \rightarrow \mathbf{Prop}). \diamond^{\hat{\kappa}_2} ((C \times R) \times \{*\}) P_2'$
2. $P_2' = \square^{\hat{\kappa}_3} ((C \times R) \times \{*\}) P_2''$

Example - Scheduler

Introduction

Preliminaries

Modal Operators

Modal Fixed-Point Operators

- Fixed-Points of Predicate Transformers
- Modal Fixed-Point Operators
- Normalforms
- The syntactic Monotonicity Criterion
- The syntactic Monotonicity Criterion 1,2
- The syntactic Monotonicity Criterion 2,3
- The syntactic Monotonicity Criterion 4
- The syntactic Monotonicity Criterion 5
- Example - Some Temporal Connectives
- Example - Scheduler
- Example - Scheduler

Conclusions

Recall Σ_M from the Signature for the Scheduler

	TX	
<i>request</i>	$(C \rightarrow X) \times$	$\hat{\kappa}_1 \cdots$
<i>assign</i>	$((C \times R) \rightarrow X) \times$	$\hat{\kappa}_2 \cdots$
<i>finish</i>	$((C \times R) \rightarrow X) \times$	$\hat{\kappa}_3 \cdots$
<i>get_resource</i>	$(R + 1)$	$\hat{\kappa}_4 \cdots$

The following Property describes that

1. after a Resource is assigned to a Client
2. and after any Client returns a Resource
3. a resource is available and the game loops forever

1. $P_2 = \mu(Z : \mathbf{Self} \rightarrow \mathbf{Prop}). \diamond^{\hat{\kappa}_2} ((C \times R) \times \{*\}) P'_2$
2. $P'_2 = \square^{\hat{\kappa}_3} ((C \times R) \times \{*\}) P''_2$
3. $P''_2 = \lambda(x : \mathbf{Self}). up?(get_resource\ x) \wedge (Z\ x)$

Conclusions

Introduction

Preliminaries

Modal Operators

Modal Fixed-Point Operators

Conclusions

● Conclusions

- Single-Step Modal Operators finer than Infinitary Modal Operators
- Modal Fixed-Point Operators are helpful for defining Finiteness Properties
- Still complicated, unhandy Syntax

Introduction

Preliminaries

Modal Operators

Modal Fixed-Point Operators

Conclusions

● Conclusions

●

Thank you very much for your attention!