Checking multi-view consistency of discrete systems with respect to periodic sampling abstractions

Maria Pittou[1] and Stavros Tripakis[2]

[1]Aalto University

[2]Aalto University and UC Berkeley

CL Day, Aalto 2016

Outline

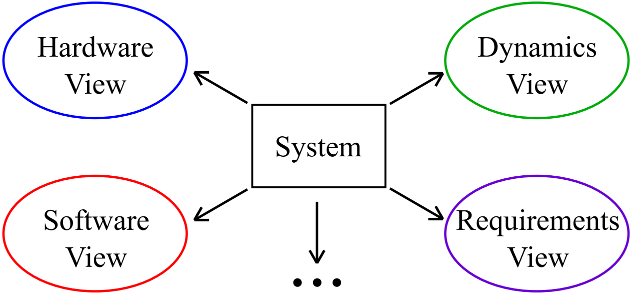
1. Introduction
   - Motivation for multi-view modeling
   - Related work
   - System, views, view consistency
2. Contribution
3. Formal framework
   - Discrete systems
   - Periodic samplings
4. Detecting view inconsistency
   - The multi-view consistency problem(s)
   - Algorithm for checking view inconsistencies
5. Conclusions and Future work

Systems are complex and large, hence their modeling involves multiple design teams.

Motivation

Multi-view modeling (MVM)

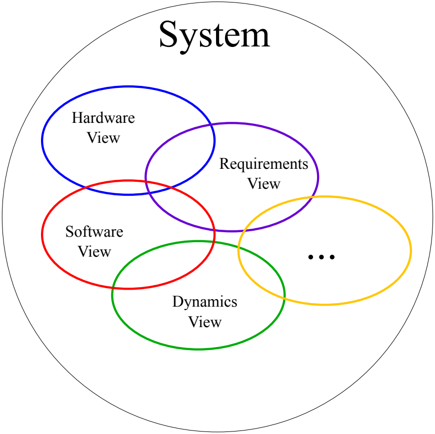
The stakeholders engaged in the modeling of a system, obtain seperate views of the system.



Motivation

Multi-view consistency

One of the main challenges in multi-view modeling is to ensure consistency among the different views.



Related work

Specific view consistency problems

- Vooduu: Verification of Object-Oriented Designs Using UPPAAL, 2004, K. Diethers and M. Huhn
- Semantically Configurable Consistency Analysis for Class and Object Diagrams, 2011, Maoz et al

Formal framework for MVM

- Basic problems in multi-view modeling, 2014, J. Reineke and S. Tripakis.
- Basic problems in multi-view modeling, 2016 (journal version), J. Reineke, C. Stergiou and S. Tripakis.

Problem to be solved

The multi-view consistency problem (informally)

Given a (finite) set of views, are they consistent?

$\downarrow$

1) How are the views (and the system) described?

2) How are the views derived from the system?

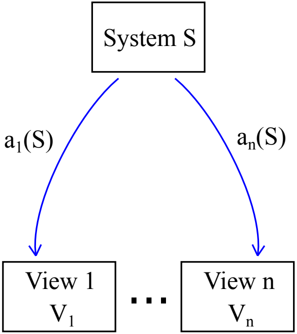
3) What does view consistency mean?

Outline

1. Introduction
   - Motivation for multi-view modeling
   - Related work
   - System, views, view consistency
2. Contribution
3. Formal framework
   - Discrete systems
   - Periodic samplings
4. Detecting view inconsistency
   - The multi-view consistency problem(s)
   - Algorithm for checking view inconsistencies
5. Conclusions and Future work

System, views, and abstraction functions

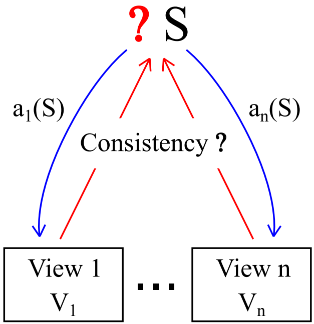
- System S: set of behaviors
- View V: set of behaviors
- Abstraction function $V = a(S)$



View consistency

View consistency

The views $V_1, \ldots, V_n$ are *consistent with respect to the abstraction functions* $a_1, \ldots, a_n$, if there exists a system S so that $V_1 = a_1(S), \ldots, V_n = a_n(S)$.



- We call such a system S a *witness system* to the consistency of V_1 and V_2.
- If there is no such system, then we conclude that the views are inconsistent.

Outline

1. Introduction
   - Motivation for multi-view modeling
   - Related work
   - System, views, view consistency
2. Contribution
3. Formal framework
   - Discrete systems
   - Periodic samplings
4. Detecting view inconsistency
   - The multi-view consistency problem(s)
   - Algorithm for checking view inconsistencies
5. Conclusions and Future work

Previous work vs current work

The multi-view consistency problem

1) Basic problems in multi-view modeling, 2014

   $\rightarrow$ **System, Views:** discrete systems (transition systems)

   $\rightarrow$ **Abstraction functions:** projections of state variables

2) Journal version 2016

   $\rightarrow$ **System, Views**: finite automata

   $\rightarrow$ **Abstraction functions:** projections of an alphabet of events onto a subalphabet.

Previous work vs current work

The multi-view consistency problem

1) Basic problems in multi-view modeling, 2014

→ **System, Views:** discrete systems (transition systems)

→ **Abstraction functions:** projections of state variables

2) Journal version 2016

→ **System, Views**: finite automata

→ **Abstraction functions:** projections of an alphabet of events onto a subalphabet.

- Current work

  → **System, Views:** discrete systems (transition systems)

  → **Abstraction functions:** periodic samplings

Outline

1. Introduction
   - Motivation for multi-view modeling
   - Related work
   - System, views, view consistency
2. Contribution
3. Formal framework
   - Discrete systems
   - Periodic samplings
4. Detecting view inconsistency
   - The multi-view consistency problem(s)
   - Algorithm for checking view inconsistencies
5. Conclusions and Future work

Symbolic discrete systems

Semantics

- **State variables:** X

$\rightarrow X = \{x, y\}$

Symbolic discrete systems

Semantics

- **State variables:** X

$\rightarrow X = \{x, y\}$

- **State:** $s : X \rightarrow \{0, 1\}$

$\rightarrow s_1 = (0, 0), s_2 = (0, 1), s_3 = (1, 0), s_4 = (1, 1)$

- **State variables:** X

$\rightarrow X = \{x, y\}$

- **State:** $s : X \rightarrow \{0, 1\}$

$\rightarrow s_1 = (0,0), s_2 = (0,1), s_3 = (1,0), s_4 = (1,1)$

- **Behavior:** finite/infinite sequence of states

$\rightarrow \sigma_1 = s_4 s_4 s_4 s_4 \cdots, \sigma_2 = s_4 s_2 s_3 s_4 \cdots,$

Symbolic discrete systems

Semantics

- **State variables:** X

$\rightarrow X = \{x, y\}$

- **State:** $s : X \rightarrow \{0, 1\}$

$\rightarrow s_1 = (0, 0), s_2 = (0, 1), s_3 = (1, 0), s_4 = (1, 1)$

- **Behavior:** finite/infinite sequence of states

$\rightarrow \sigma_1 = s_4 s_2 s_3 s_4 \cdots , \sigma_2 = s_4 s_2 s_4 s_4 \cdots ,$

- **Discrete system:** set of behaviors

$\rightarrow S = \{\sigma_1, \sigma_2, \cdots\}$

- FOS: Fully-observable discrete system

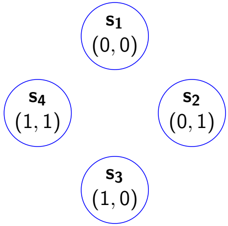
$\rightarrow$ All variables are observable

- nFOS: Non-Fully-observable discrete system

$\rightarrow$ Some variables are unobservable

Fully observable symbolic discrete system (FOS): $S = \{X, \theta, \phi\}$

- $X = \{x, y\}$

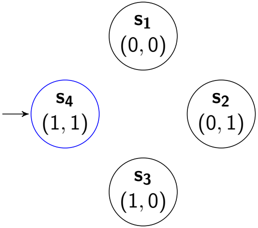


Symbolic discrete systems (FOS)

Syntax

Fully-observable symbolic discrete system (FOS): $S = \{X, \theta, \phi\}$

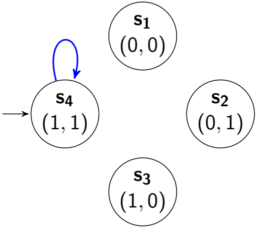
- $X = \{x, y\}$
- $\theta = x \wedge y$



Symbolic discrete systems (FOS)
Syntax

Fully observable symbolic discrete system (FOS): $S = \{X, \theta, \phi\}$

- $X = \{x, y\}$
- $\theta = x \wedge y$
- $\phi = (x \wedge y \rightarrow x' \wedge y')$

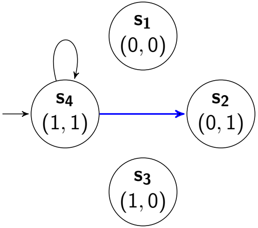


Symbolic discrete systems (FOS)

Syntax

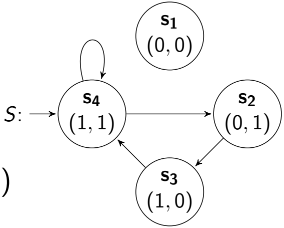
Fully observable symbolic discrete system (FOS): $S = \{X, \theta, \phi\}$

- $X = \{x, y\}$
- $\theta = x \wedge y$
- $\phi = (x \wedge y \rightarrow x' \wedge y') \wedge (x \wedge y \rightarrow \neg x' \wedge y')$



Fully observable symbolic discrete system (FOS): $S = \{X, \theta, \phi\}$

- $X = \{x, y\}$
- $\theta = x \wedge y$
- $\phi = (x \wedge y \rightarrow x' \wedge y') \wedge (x \wedge y \rightarrow \neg x' \wedge y') \wedge (\neg x \wedge y \rightarrow x' \wedge \neg y')$



Non-fully-observable symbolic discrete system (nFOS): $S = \{X, Z, \theta, \phi\}$

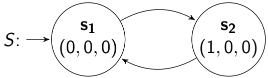
- $X = \{x, y\} \leftarrow$ **observable**
- $Z = \{z\} \leftarrow$ **unobservable**

Symbolic discrete systems (nFOS)

Syntax

Non-fully-observable symbolic discrete system (nFOS): $S = \{X, Z, \theta, \phi\}$

- $X = \{x, y\} \leftarrow$ **observable**
- $Z = \{z\} \leftarrow$ **unobservable**
- $s : X \cup Z \rightarrow \{0, 1\}$
- $\theta = \neg x \wedge \neg y \neg z$
- $\phi = (\neg x \wedge \neg y \wedge \neg z \rightarrow x' \wedge y' \wedge \neg z') \wedge$
  $\wedge (x \wedge \neg y \wedge \neg z \rightarrow \neg x' \wedge \neg y' \wedge \neg z')$



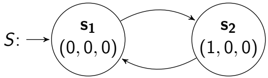
Symbolic discrete systems (nFOS)

Syntax

Non-fully-observable symbolic discrete system (nFOS): $S = \{X, Z, \theta, \phi\}$

- Observable behavior

$$\boxed{\sigma = (0,0)(1,0)(0,0)(1,0)\cdots}$$



- Unobservable behavior

$$\sigma = (0,0,0)(1,0,0)(0,0,0)(1,0,0)\cdots$$

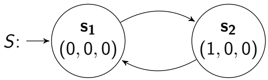
▷ Every FOS is a special case of nFOS with $Z = \emptyset$.

Symbolic discrete systems (nFOS)
Syntax

Non-fully-observable symbolic discrete system (nFOS): $S = \{X, Z, \theta, \phi\}$

- Observable behavior

$$\sigma = (0,0)(1,0)(0,0)(1,0)\cdots$$



- Unobservable behavior

$$\boxed{\sigma = (0,0,0)(1,0,0)(0,0,0)(1,0,0)\cdots}$$

▷ Every FOS is a special case of nFOS with $Z = \emptyset$.

Periodic sampling abstraction functions

Example and Closure

Behavior:	σ	$=$	$s_0 s_1 \underline{s_2} s_3 s_4 \underline{s_5} s_6 s_7 \underline{s_8} \cdots$
			$\downarrow \quad \downarrow \quad \downarrow$
Periodic sampling: $T = 3$, period $\tau = 2$, initial position	$a_{T,\tau}(\sigma)$	$=$	$s_2 \quad s_5 \quad s_8 \cdots$

Periodic sampling abstraction functions

Example and Closure

Behavior: $\sigma = s_0 s_1 \underline{s_2} s_3 s_4 \underline{s_5} s_6 s_7 \underline{s_8} \cdots$

$$\downarrow \quad \downarrow \quad \downarrow$$

Periodic sampling:
$T = 3$, period
$\tau = 2$, initial position

$a_{T,\tau}(\sigma) = s_2 \quad s_5 \quad s_8 \cdots$

Theorem

nFOS (and FOS) are closed under periodic sampling

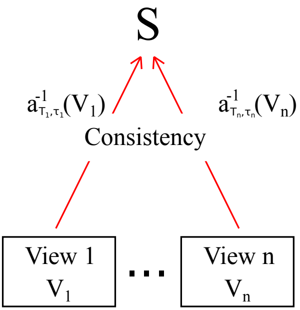
Periodic sampling abstraction functions

Framework

System	Abstraction function	View
$\downarrow$	$\downarrow$	$\downarrow$
nFOS	periodic sampling	nFOS
S	$\xrightarrow{a_{T,\tau}}$	$V = a_{T,\tau}(S)$

*Inverse periodic samplings

- Inverse periodic samplings $a^{-1}_{T,\tau}$



→ FOS are **NOT** closed under inverse periodic samplings

→ nFOS are closed under inverse periodic samplings

Outline

1. Introduction
   - Motivation for multi-view modeling
   - Related work
   - System, views, view consistency
2. Contribution
3. Formal framework
   - Discrete systems
   - Periodic samplings
4. Detecting view inconsistency
   - The multi-view consistency problem(s)
   - Algorithm for checking view inconsistencies
5. Conclusions and Future work

- We set $\tau = 0$ for the rest of the presentation.

Problems

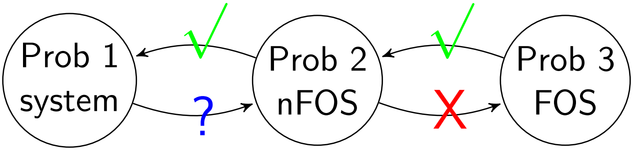
Given a finite set of nFOS $V_i = (X, W_i, \theta_i, \phi_i)$ and periodic samplings a_{T_i}, for $1 \leq i \leq n$, check whether:

(1) there exists a system (set of behaviors) S

(2) there exists an nFOS S

(3) there exists a FOS S

such that $a_{T_i}(S) = V_i$ for every $1 \leq i \leq n$.



Problems

Given a finite set of nFOS $V_i = (X, W_i, \theta_i, \phi_i)$ and periodic samplings a_{T_i}, for $1 \leq i \leq n$, check whether:

(1) there exists a system (set of behaviors) S

(2) there exists an nFOS S

(3) there exists a FOS S

such that $a_{T_i}(S) = V_i$ for every $1 \leq i \leq n$.

Outline

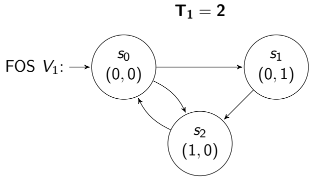
1. Introduction
   - Motivation for multi-view modeling
   - Related work
   - System, views, view consistency
2. Contribution
3. Formal framework
   - Discrete systems
   - Periodic samplings
4. Detecting view inconsistency
   - The multi-view consistency problem(s)
   - Algorithm for checking view inconsistencies
5. Conclusions and Future work

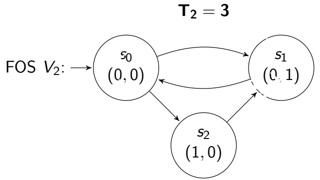
Detecting view inconsistency

Intuition

- What does view inconsistency mean for our framework?

$\rightarrow$ The views return a different set of states at the critical positions (of the behaviors of a candidate witness system)



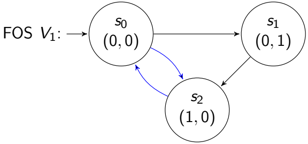


Detecting view inconsistency
Intuition

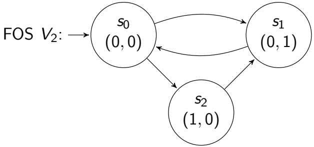
- What does view inconsistency mean for our framework?

→ The views return a different set of states at the critical positions (of the behaviors of a candidate witness system)

$\mathbf{T_1 = 2}$



$\mathbf{T_2 = 3}$



Positions in the witness system

0	1	2	3	4	5	6
↓		↓		↓		↓
s_0	*	s_2	*	s_0	*	s_2

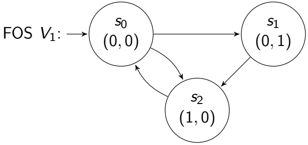
Detecting view inconsistency

Intuition

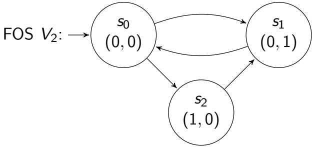
- What does view inconsistency mean for our framework?

$\rightarrow$ The views return a different set of states at the critical positions (of the behaviors of a candidate witness system)

$\mathbf{T_1 = 2}$



$\mathbf{T_2 = 3}$



Positions in the witness system

0	1	2	3	4	5	6
$\downarrow$		$\downarrow$		$\downarrow$		$\downarrow$
s_0	$*$	s_2	$*$	s_0	$*$	s_2
$\downarrow$			$\downarrow$			$\downarrow$
$\mathbf{s_0}$			$*$			$\mathbf{s_2}$ X

$\looparrowright$ The views are inconsistent.

- The algorithm applies to sets of views where: (i) every view generates only infinite behaviors or (ii) every view generates only finite behaviors behaviors.

Intuition for the algorithm:

Given some views described by nFOS (or FOS) and obtained from periodic samplings:

(1) Consider a special construction that encodes the "critical" positions.

(2) Obtain the "composition" of modified versions of views with the construction of (1).

(3) Apply state-based reachability to check for inconsistencies and if **YES** report inconsistency and if **NO** report inconclusive.

View inconsistency algorithm
Soundness

- The algorithm is sound i.e., if it reports inconsistency then the views are indeed inconsistent.

Theorem

If the algorithm reports inconsistency then there exists no solution to Problems 1,2, and 3.

$\rightarrow$ Problem 1: semantic witness system

$\rightarrow$ Problem 2: nFOS witness system

$\rightarrow$ Problem 3: FOS witness system

View inconsistency algorithm

Completeness

- The algorithm is NOT complete, i.e., if the algorithm reports inconclusive then the views can either be consistent or not.
- The algorithm relies on a state-based reachability, hence it neglects inconsistencies that involve the transition structure as well.

Outline

1. Introduction
   - Motivation for multi-view modeling
   - Related work
   - System, views, view consistency
2. Contribution
3. Formal framework
   - Discrete systems
   - Periodic samplings
4. Detecting view inconsistency
   - The multi-view consistency problem(s)
   - Algorithm for checking view inconsistencies
5. Conclusions and Future work

Recap

- Notions of (forward and inverse) periodic sampling abstraction functions.
- Closure of discrete systems under these abstraction functions.
- Study of multi-view consistency problem for discrete systems in the periodic sampling setting.

  $\mapsto$ A sound but not complete algorithm for detecting inconsistencies.

Future work

- Develop a complete view consistency algorithm.
- Consider other abstraction functions than projections or periodic samplings.
- Heterogeneous instantiations of the multi-view modeling framework.
- Experimentation with case studies.

Thank you! . . . Questions?

- Let X be a finite set of variables.
- $a_{T,\tau}$ denotes a periodic sampling abstraction function from $\mathcal{U}(X)$ to $\mathcal{D}(X)$ w.r.t. period T and initial position τ

Definition of forward periodic sampling

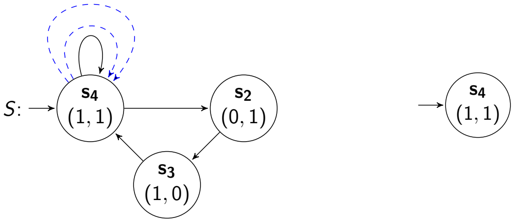
A periodic sampling abstraction function $a_{T,\tau} : \mathcal{U}(X) \to \mathcal{D}(X)$ is defined such that for every behavior $\sigma = s_0 s_1 \cdots \in \mathcal{U}(X)$, $a_{T,\tau}(\sigma) := s'_0 s'_1 \cdots \in \mathcal{D}(X)$ where $s'_i = s_{\tau+i\cdot T}$ for every $i \geq 0$.

- For a system $\mathcal{S} \subseteq \mathcal{U}(X)$, we define $a_{T,\tau}(\mathcal{S}) := \{a_{T,\tau}(\sigma) \mid \sigma \in \mathcal{S}\}$.

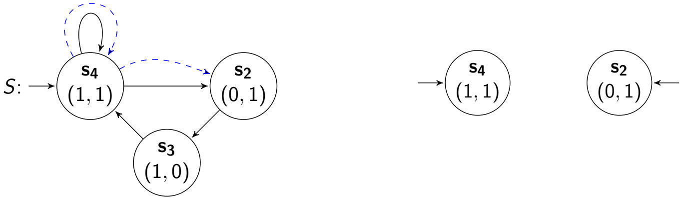
Back up slides

Periodic sampling abstraction functions (Example on FOS)

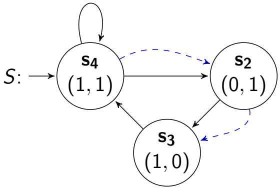
- Apply periodic sampling $a_{T,\tau}$ with $T = 3$ and $\tau = 2$ to FOS S

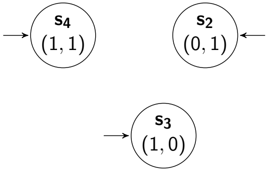


- Apply periodic sampling $a_{T,\tau}$ with $T = 3$ and $\tau = 2$ to FOS S



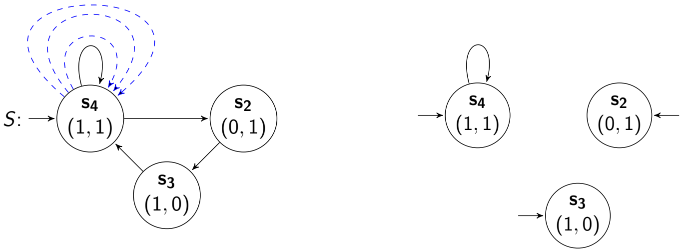
- Apply periodic sampling $a_{T,\tau}$ with $T = 3$ and $\tau = 2$ to FOS S



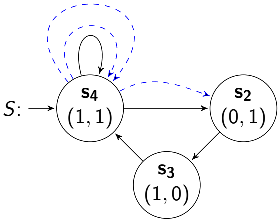


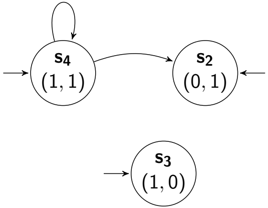
Periodic sampling abstraction functions (Example on FOS)

- Apply periodic sampling $a_{T,\tau}$ with $T = 3$ and $\tau = 2$ to FOS S



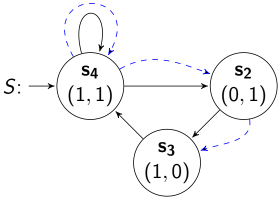
- Apply periodic sampling $a_{T,\tau}$ with $T = 3$ and $\tau = 2$ to FOS S

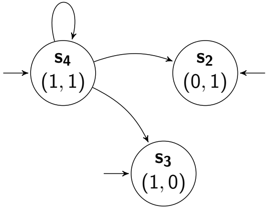




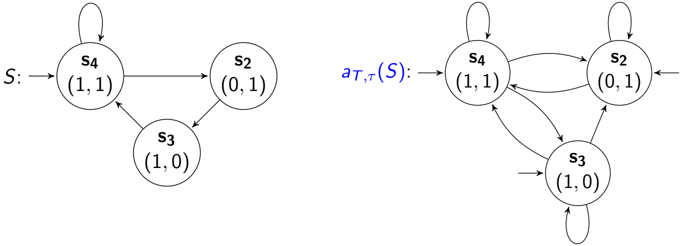
Periodic sampling abstraction functions (Example on FOS)

- Apply periodic sampling $a_{T,\tau}$ with $T = 3$ and $\tau = 2$ to FOS S





- Apply periodic sampling $a_{T,\tau}$ with $T = 3$ and $\tau = 2$ to FOS S



$\rightarrow$ $a_{T,\tau}(S)$ is a view of S obtained by $a_{T=3,\tau=2}$

Theorem

Given a FOS system $S = (X, \theta, \phi)$ and periodic sampling $a_{T,\tau}$, there exists a FOS system S' such that $[\![S']\!] = a_{T,\tau}([\![S]\!])$.

Proof.

We define the FOS $S' = (X, \theta, \phi')$, where θ' contains all states over X which can be reached from some initial state of S in exactly τ steps; and ϕ' is defined as follows. Let s, s' be two states over X. Then $\phi'(s, s')$ iff S has a path from s to s' of length exactly T. $\square$

Theorem

Given a FOS system $S = (X, \theta, \phi)$ and periodic sampling $a_{T,\tau}$, there exists a FOS system S' such that $[\![S']\!] = a_{T,\tau}([\![S]\!])$.

Proof.

Consider an arbitrary behavior $\sigma = s_0 s_1 s_2 \cdots \in [\![S]\!]$. Applying the periodic sampling $a_{T,\tau}$ to σ we obtain the behavior $a_{T,\tau}(\sigma) = s_\tau s_{\tau+T} s_{\tau+2T} \cdots$. By construction of S' we have that $\theta'(s_\tau)$ and $\phi'(s_{\tau+iT}, s_{\tau+(i+1)T})$ for every $i \geq 0$, which implies that $a_{T,\tau}(\sigma) \in [\![S']\!]$. Hence, $a_{T,\tau}([\![S]\!]) = \{a_{T,\tau}(\sigma) \mid \sigma \in [\![S]\!]\} \subseteq [\![S']\!]$.

□

Theorem

Given a FOS system $S = (X, \theta, \phi)$ and periodic sampling $a_{T,\tau}$, there exists a FOS system S' such that $[\![S']\!] = a_{T,\tau}([\![S]\!])$.

Proof.

Conversely, let $\sigma' = s'_0 s'_1 s'_2 \ldots \in [\![S']\!]$. Since $\phi'(s'_0)$, by definition of S' there exists a state s_0 in S with $\theta(s_0)$ so that s'_0 can be reached from s_0 in exactly τ steps. Moreover, for σ' we have that $\phi'(s'_i, s'_{i+1})$, thus there exists a path in S from s'_i to s'_{i+1} of length exactly T for every $i \geq 0$. Then, we obtain the behavior $\sigma = s_0 s_1 s_2 \cdots \in [\![S]\!]$ where $s_{\tau+iT} = s'_i$ for every $i \geq 0$. Hence, $a_{T,\tau}(\sigma) \in a_{T,\tau}([\![S]\!])$ and $[\![S']\!] \subseteq a_{T,\tau}([\![S]\!])$ which completes our proof. □

Theorem

Given a nFOS system $S = (X, Z, \theta, \phi)$ and periodic sampling $a_{T,\tau}$, there exists a nFOS system S' such that $[\![S']\!] = a_{T,\tau}([\![S]\!])$.

- Let X be a finite set of variables.
- $a^{-1}_{T,\tau}$ denotes an inverse periodic sampling abstraction function from $\mathcal{D}(X)$ to $\mathcal{U}(X)$ w.r.t. period T and initial position τ

Definition of inverse periodic sampling

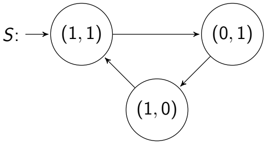
An inverse periodic sampling $a^{-1}_{T,\tau} : \mathcal{D}(X) \to \mathcal{U}(X)$ is defined by the mapping $a^{-1}_{T,\tau} : \mathcal{D}(X) \to \mathcal{U}(X)$ such that for every behavior $\sigma = s_0 s_1 \cdots \in \mathcal{D}(X)$, $a^{-1}_{T,\tau}(\sigma) := \{\sigma' \mid \sigma' = s'_0 s'_1 \cdots \in \mathcal{U}(X)$ s.t. $s'_{\tau+i\cdot T} = s_i, i \geq 0\}$ or equivalently $a^{-1}_{T,\tau}(\sigma) := \{\sigma' \mid a_{T,\tau}(\sigma') = \sigma\}$.

- For a system $\mathcal{S} \subseteq \mathcal{U}(X)$, we define $a^{-1}_{T,\tau}(\mathcal{S}) := \bigcup_{\sigma \in \mathcal{S}} a^{-1}_{T,\tau}(\sigma)$.

Back up slides

Non closure of FOS under inverse periodic periodic sampling

- Consider the FOS $S = (\{x, y\}, \theta, \phi)$ obtained with periodic sampling a_T and period $T = 2$.



Position	σ
$i = 0$	(1,1)
$i = 1$	?
$i = 2$	(0,1)
$i = 3$	?
$i = 4$	(1,0)
$i = 5$	?
$i = 6$	(1,1)
$i = 7$	$\vdots$

$\rightarrow$The first 6 states in the unique behavior σ of S' should be distinct.
Yet, this is not possible, since we only have two Boolean variables x, y.

Theorem

Given a system $S = (X, Z, \theta, \phi)$ and inverse periodic sampling $a^{-1}_{T,\tau} : \mathcal{D}(X \cup Z) \to \mathcal{U}(X \cup Z \cup W)$, there exists always a non-fully-observable system $S' = (X \cup Z, W, \theta', \phi')$ such that $[\![S']\!] = a^{-1}_{T,\tau}([\![S]\!])$.

Proof.

Given the nFOS $S = (X, Z, \theta, \phi)$ let R denote the set of reachable states of S over $X \cup Z$. Moreover, let $|R| = n$ and consider a set of Boolean variables W such that $|W| \geq \lfloor \log_2(n \cdot (T-1) + \tau) \rfloor$ (here we assume that $T \geq 2$; if $T = 1$ then we can simply take $S' = S$). By definition we have that $\sigma \in a^{-1}_{T,\tau}([\![S]\!])$ iff $a_{T,\tau}(\sigma) \in [\![S]\!]$. Moreover, $\sigma' = a_{T,\tau}(\sigma) = s_\tau s_{\tau+T} s_{\tau+2T} \cdots$, i.e., each behavior σ' in $[\![S]\!]$ has been obtained with starting position τ and period T. $\square$

Back up slides

Closure of nFOS under inverse periodic sampling- proof-2

Theorem

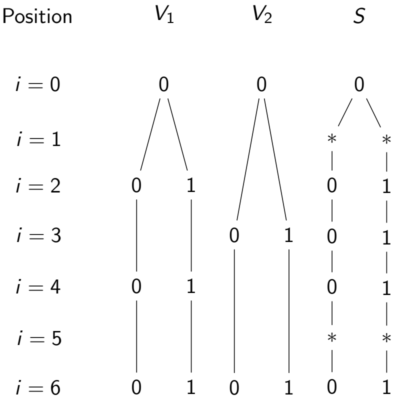
Given a system $S = (X, Z, \theta, \phi)$ *and inverse periodic sampling* $a^{-1}_{T,\tau} : \mathcal{D}(X \cup Z) \rightarrow \mathcal{U}(X \cup Z \cup W)$, *there exists always a non-fully-observable system* $S' = (X \cup Z, W, \theta', \phi')$ *such that* $[\![S']\!] = a^{-1}_{T,\tau}([\![S]\!])$.

Proof.

The system S' has to be defined such that each behavior in $[\![S']\!]$ results from σ' by (i) adding τ transitions (or states) in the beginning of σ' and T transitions (or $T-1$ states) in between the transition $\phi(s_{\tau+iT}, s_{\tau+(i+1)T})$ for every $i \geq 0$, and by (ii) replacing each $s_{\tau+iT}$ in σ' with $s'_{\tau+iT} = h_{X\cup Z}(s_{\tau+iT})$. Since S consists of n reachable states then S' should have at least $n(T-1)+\tau$ more reachable states or equivalently $\lfloor \log_2(n \cdot (T-1)+\tau) \rfloor$ more Boolean variables. One can then obtain a nFOS S' over $X \cup Z \cup W$, where $X \cup Z$ and W denote the set of observable and unobservable variables respectively, such that $[\![S']\!] = a^{-1}_{T,\tau}([\![S]\!])$. $\square$

Counterexample

The views V_1 and V_2 have been sampled with periods $T_1 = 2$ and $T_2 = 3$ respectively. The observable behavior of the views is shown in the form of trees. There exists a nFOS system S witness to the consistency of V_1 and V_2. However, there does not exist any fully-observable system with a single state variable x.



Back up slides

Main lemma-proof-1

- $Y_i^m = \{s_i \mid s_i : X \to \mathbb{B}$ occurs at position m in some behavior $\sigma \in [\![S_i]\!]\}$, where S_i is a nFOS for every $1 \leq i \leq n$.

Lemma

Consider a set of views $S_1, ..., S_n$ and periodic samplings a_{T_i}, for $1 \leq i \leq n$. If there exist $i, j \in \{1, ..., n\}$ and positive integer m multiple of $LCM(T_i, T_j)$ such that $Y_j^{m/T_j} \neq Y_i^{m/T_i}$, then $S_1, ..., S_n$ are inconsistent.

Proof.

Let $S_i = (X, W_i, \theta_i, \phi_i)$. Assume that there exist $i, j \in \{1, ..., n\}$ and positive integer m multiple of $LCM(T_i, T_j)$ such that $Y_j^{m/T_j} \neq Y_i^{m/T_i}$. W.l.o.g., suppose that there exists a state $s \in Y_i^{m/T_i} \setminus Y_j^{m/T_j}$. We would like to prove that the views $S_1, ..., S_n$ are inconsistent. Assume to the contrary that they are consistent.

$\square$

- $Y_i^m = \{s_i \mid s_i : X \to \mathbb{B}$ occurs at position m in some behavior $\sigma \in [\![S_i]\!]\}$, where S_i is a nFOS for every $1 \leq i \leq n$.

Lemma

Consider a set of views $S_1, \ldots, S_n$ and periodic samplings a_{T_i}, for $1 \leq i \leq n$. If there exist $i, j \in \{1, \ldots, n\}$ and positive integer m multiple of $LCM(T_i, T_j)$ such that $Y_j^{m/T_j} \neq Y_i^{m/T_i}$, then $S_1, \ldots, S_n$ are inconsistent.

Proof.

This implies that there exists a system $\mathcal{S}$ over $\mathcal{U}(X)$ such that $a_{T_k}(\mathcal{S}) = [\![S_k]\!]$ for every $1 \leq k \leq n$. Then, $a_{T_i}(\mathcal{S}) = [\![S_i]\!]$ and $a_{T_j}(\mathcal{S}) = [\![S_j]\!]$. Since there exists state $s \in Y_i^{m/T_i} \setminus Y_j^{m/T_j}$, then there exists some behavior $\sigma_i \in [\![S_i]\!]$ such that σ_i is at position m/T_i at state s. Because $a_{T_i}(\mathcal{S}) = [\![S_i]\!]$ we have that $\sigma_i \in a_{T_i}(\mathcal{S})$. By definition, $a_{T_i}(\mathcal{S}) = \{a_{T_i}(\sigma) \mid \sigma \in \mathcal{S}\}$ and because $\sigma_i \in a_{T_i}(\mathcal{S})$ then $\exists \sigma \in \mathcal{S}$ such that $a_{T_i}(\sigma) = \sigma_i$.

□

- $Y_i^m = \{s_i \mid s_i : X \to \mathbb{B}$ occurs at position m in some behavior $\sigma \in [\![S_i]\!]\}$, where S_i is a nFOS for every $1 \leq i \leq n$.

Lemma

Consider a set of views $S_1, ..., S_n$ and periodic samplings a_{T_i}, for $1 \leq i \leq n$. If there exist $i, j \in \{1, ..., n\}$ and positive integer m multiple of $LCM(T_i, T_j)$ such that $Y_j^{m/T_j} \neq Y_i^{m/T_i}$, then $S_1, ..., S_n$ are inconsistent.

Proof.

By construction, σ is at state s at position m. Since $\sigma \in \mathcal{S}$ we have that $a_{T_j}(\sigma) \in a_{T_j}(\mathcal{S}) = [\![S_j]\!]$. Let $\sigma_j = a_{T_j}(\sigma)$. Because σ is at state s at position m, σ_j must be at the same state s at position m/T_j. This in turn implies that $s \in Y_j^m$, which is a contradiction.

$\square$

View inconsistency algorithm-1

- Consider a finite set of views defined by the nFOS $S_i = (X, W_i, \theta_i, \phi_i)$, and obtained by applying some periodic sampling a_{T_i} with sampling period T_i, for $i = 1, \ldots, n$, respectively.
- Let $T = LCM(T_1, \ldots, T_n)$, $P = \mathcal{P}(\{p_{T_1}, \ldots, p_{T_n}\})$ and $M = \{0, m_1, \ldots, m_k\}$ denote respectively the hyper-period of periods, the labels of periods, and the ordered set of multiples of periods up to their hyper-period.

View inconsistency algorithm-2

Steps of the algorithm for detecting inconsistency among the views $S_1, \ldots, S_n$:

- **Step 1:** Construct for each S_i, $i = 1, \ldots, n$, the FA $L_i = (Q_i, \Sigma_i, Q_{i_0}, \Delta_i, F_i)$ where $Q_i = \mathbb{B}^{X \cup W_i}$, $\Sigma_i = \{p_{T_i}\}$, $Q_{i_0} = \{s \mid \theta_i(s)\}$, $F_i = \emptyset$, and $\Delta_i \subseteq Q_i \times \Sigma_i \times Q_i$ is defined such that $(s, p_{T_i}, s') \in \Delta_i$ iff $\phi_i(s, s')$.
- **Step 2:** Determinize each of the FA L_i and obtain the equivalent deterministic FA dL_i for every $i = 1, \ldots, n$.
- **Step 3:** Construct the hyper-period automaton H w.r.t. the periods $T_1, \ldots, T_n$.
- **Step 4:** Obtain the label-driven composition $C = (dL_1, \ldots, dL_n, H)$ w.r.t HPA H.
- **Step 5:** Let $s = (s_1, \ldots, s_n, m)$ be a state of C, and let $I_s = \{i \in \{1, ..., n\} \mid p_{T_i} \in \pi(m)\}$. The algorithm reports **inconsistency** if C contains at least one reachable state $s = (s_1, \ldots, s_n, m)$ where s_i are states of dL_i for $i = 1, \ldots, n$ respectively, and $m \in F$ is a final state of H, such that $\exists i, j \in I_s : h_X(s_i) \neq h_X(s_j)$. Otherwise, it reports **inconclusive.**

View inconsistency algorithm-example

Hyper-period automaton (HPA) example

(1) Consider a special construction that encodes the "critical" positions.

- Two views V_1 and V_2 with periods $T_1 = 2$ and $T_2 = 3$
- $LCM(T_1, T_2) = 6$
- $M = \{0, 2, 3, 4\}$

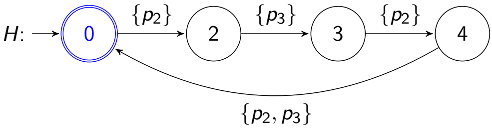
View inconsistency algorithm-example

Hyper-period automaton (HPA) example

(1) Consider a special construction that encodes the "critical" positions.

- Two views V_1 and V_2 with periods $T_1 = 2$ and $T_2 = 3$
- $LCM(T_1, T_2) = 6$
- $M = \{0, 2, 3, 4\}$

HPA w.r.t. the periods 2 and 3.



$\rightarrow$ Positions: $0, 2, 3, 4, 6, 6+2, 6+3, 6+4, 6+6$

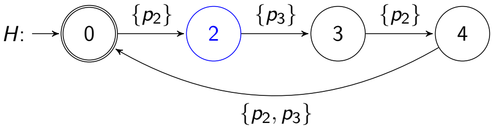
View inconsistency algorithm-example

Hyper-period automaton (HPA) example

(1) Consider a special construction that encodes the "critical" positions.

- Two views V_1 and V_2 with periods $T_1 = 2$ and $T_2 = 3$
- $LCM(T_1, T_2) = 6$
- $M = \{0, 2, 3, 4\}$

HPA w.r.t. the periods 2 and 3.



$\rightarrow$ Positions: $0, 2, 3, 4, 6, 6+2, 6+3, 6+4, 6+6$

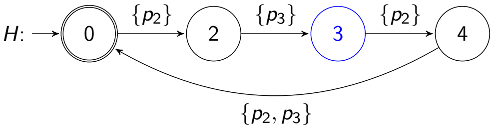
View inconsistency algorithm-example

Hyper-period automaton (HPA) example

(1) Consider a special construction that encodes the "critical" positions.

- Two views V_1 and V_2 with periods $T_1 = 2$ and $T_2 = 3$
- $LCM(T_1, T_2) = 6$
- $M = \{0, 2, 3, 4\}$

HPA w.r.t. the periods 2 and 3.



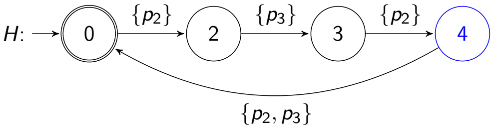
$\rightarrow$ Positions: $0, 2, 3, 4, 6, 6 + 2, 6 + 3, 6 + 4, 6 + 6$

View inconsistency algorithm-example
Hyper-period automaton (HPA) example

(1) Consider a special construction that encodes the "critical" positions.

- Two views V_1 and V_2 with periods $T_1 = 2$ and $T_2 = 3$
- $LCM(T_1, T_2) = 6$
- $M = \{0, 2, 3, 4\}$

HPA w.r.t. the periods 2 and 3.



$\rightarrow$ Positions: $0, 2, 3, 4, 6, 6+2, 6+3, 6+4, 6+6$

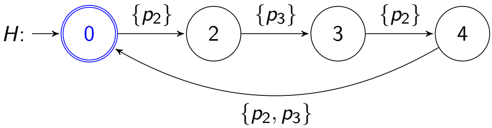
View inconsistency algorithm-example

Hyper-period automaton (HPA) example

(1) Consider a special construction that encodes the "critical" positions.

- Two views V_1 and V_2 with periods $T_1 = 2$ and $T_2 = 3$
- $LCM(T_1, T_2) = 6$
- $M = \{0, 2, 3, 4\}$

HPA w.r.t. the periods 2 and 3.



$\rightarrow$ Positions: $0, 2, 3, 4, 6, 6+2, 6+3, 6+4, 6+6$

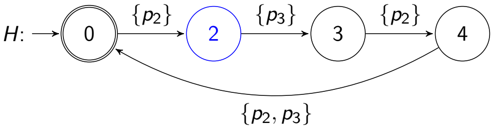
View inconsistency algorithm-example

Hyper-period automaton (HPA) example

(1) Consider a special construction that encodes the "critical" positions.

- Two views V_1 and V_2 with periods $T_1 = 2$ and $T_2 = 3$
- $LCM(T_1, T_2) = 6$
- $M = \{0, 2, 3, 4\}$

HPA w.r.t. the periods 2 and 3.



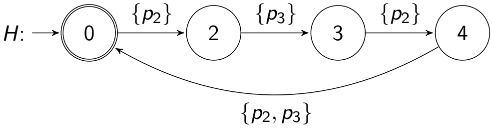
$\rightarrow$ Positions: $0, 2, 3, 4, 6, 6+2, 6+3, 6+4, 6+6$

View inconsistency algorithm-example

Hyper-period automaton (HPA) example

(1) Consider a special construction that encodes the "critical" positions.

HPA w.r.t. the periods 2 and 3.



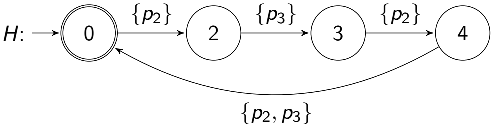
$\rightarrow$ The states encode positions: $0, 2, 3, 4, 6, 6+2, 6+3, 6+4, 6+6, \ldots$

View inconsistency algorithm-example

Hyper-period automaton (HPA) example

(1) Consider a special construction that encodes the "critical" positions.

HPA w.r.t. the periods 2 and 3.



$\rightarrow$ The states encode positions: $0, 2, 3, 4, 6, 6+2, 6+3, 6+4, 6+6, \ldots$

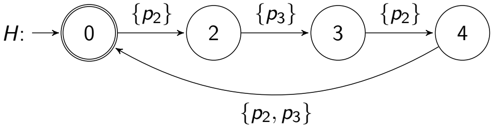
$\rightarrow$ The labels indicate the period that is active at each position.

View inconsistency algorithm-example

Hyper-period automaton (HPA) example

(1) Consider a special construction that encodes the "critical" positions.

HPA w.r.t. the periods 2 and 3.

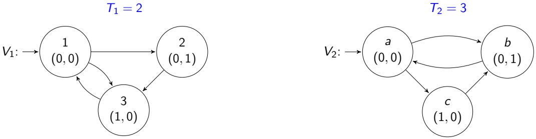


→ The states encode positions: $0, 2, 3, 4, 6, 6+2, 6+3, 6+4, 6+6, \ldots$

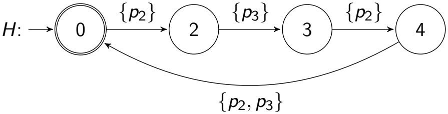
→ The labels indicate the period that is active at each position.

→ The final states are states with more than one period being active.

(2) Obtain the "composition" of the views with the HPA.



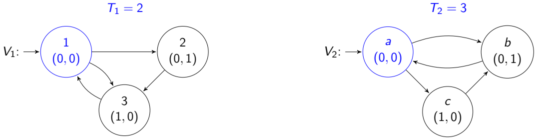
HPA w.r.t. the periods 2 and 3.



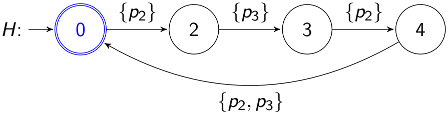
View inconsistency algorithm-example

Label driven composition example

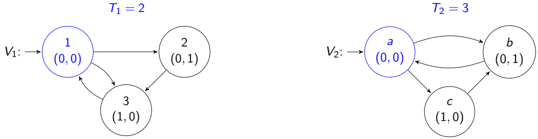
(2) Obtain the "composition" of the views with the HPA.



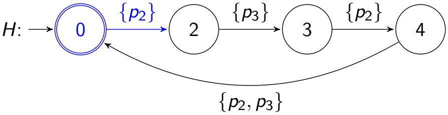
HPA w.r.t. the periods 2 and 3.



(2) Obtain the "composition" of the views with the HPA.



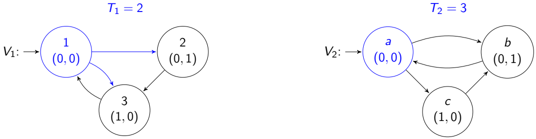
HPA w.r.t. the periods 2 and 3.



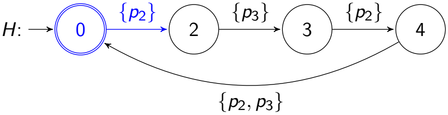
View inconsistency algorithm-example

Label driven composition example

(2) Obtain the "composition" of the views with the HPA.



HPA w.r.t. the periods 2 and 3.



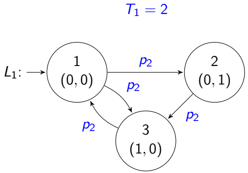
(2) Obtain the "composition" of the views with the HPA.

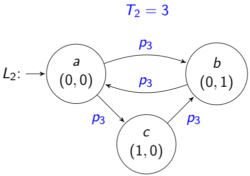
- Convert the views to finite automata labelling all their transitions with their period labels.

View inconsistency algorithm-example
Label driven composition example

(2) Obtain the "composition" of the views with the HPA.

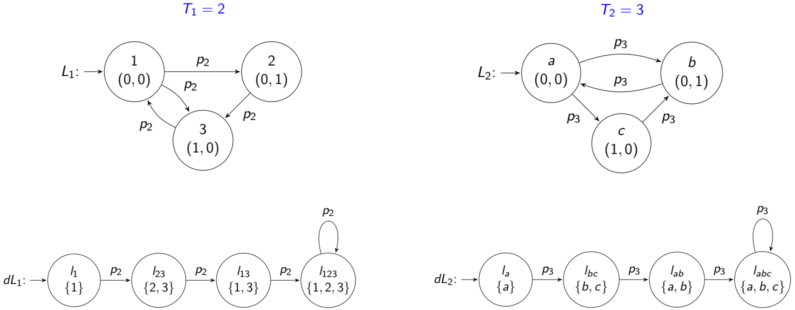
- Convert the views to finite automata labelling all their transitions with their period labels.





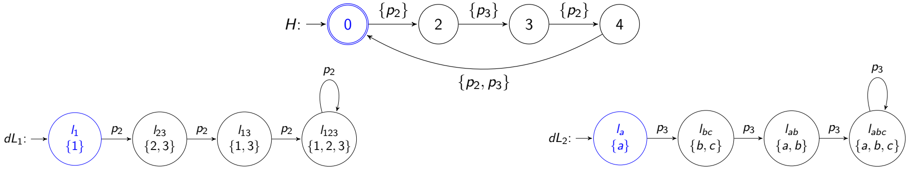
(2) Obtain the "composition" of the views with the HPA.

- Determinize the modified versions of views

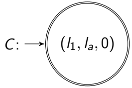


(2) Obtain the "composition" of the modified views with the HPA.

HPA w.r.t. the periods 2 and 3.



Label-driven composition

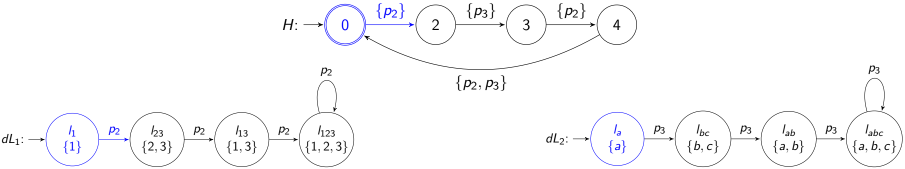


View inconsistency algorithm-example

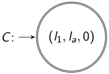
Label driven composition example

(2) Obtain the "composition" of the modified views with the HPA.

HPA w.r.t. the periods 2 and 3.

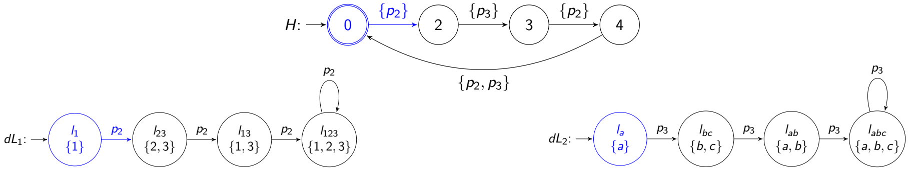


Label-driven composition

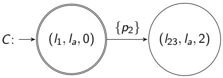


(2) Obtain the "composition" of the modified views with the HPA.

HPA w.r.t. the periods 2 and 3.



Label-driven composition

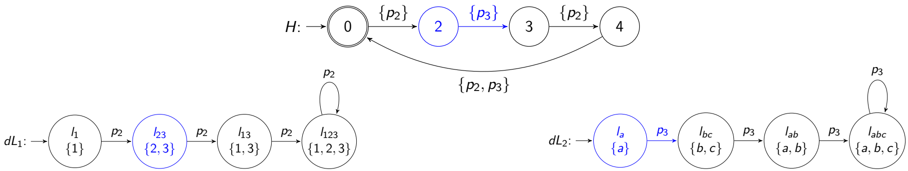


View inconsistency algorithm-example

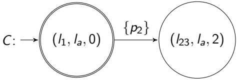
Label driven composition example

(2) Obtain the "composition" of the modified views with the HPA.

HPA w.r.t. the periods 2 and 3.



Label-driven composition

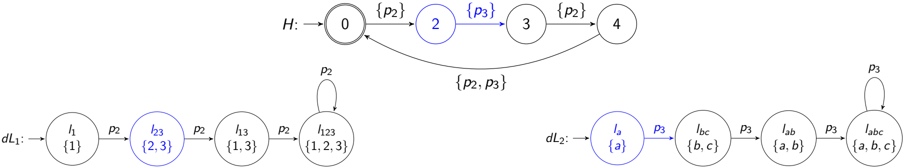


View inconsistency algorithm-example

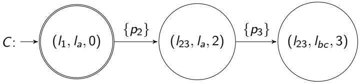
Label driven composition example

(2) Obtain the "composition" of the modified views with the HPA.

HPA w.r.t. the periods 2 and 3.

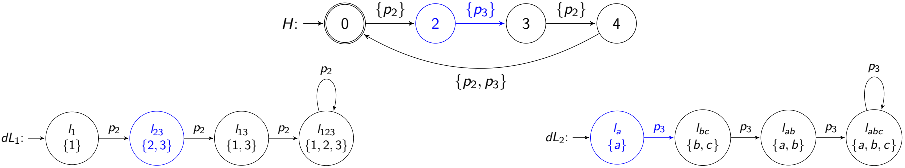


Label-driven composition

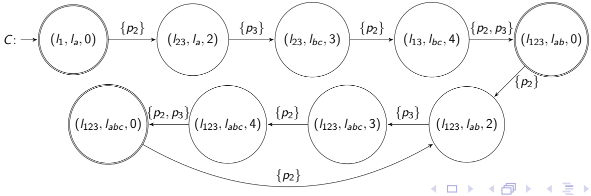


(2) Obtain the "composition" of the modified views with the HPA.

HPA w.r.t. the periods 2 and 3.

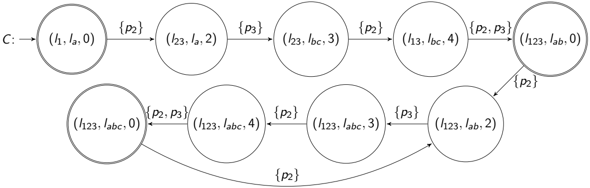


Label-driven composition

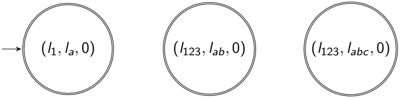


View inconsistency algorithm-example

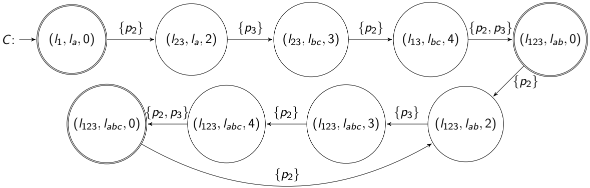
(3) Apply state-base reachability to check for inconsistencies.



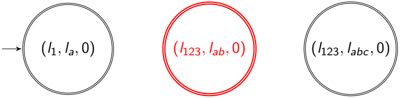
Critical states



(3) Apply state-base reachability to check for inconsistencies.



View inconsistency detected



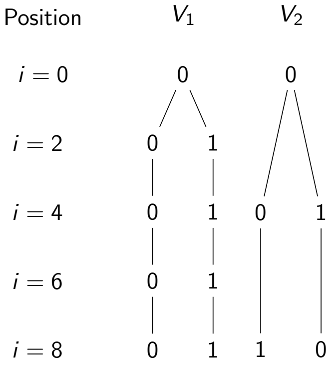
→ The algorithm reports **inconsistency** and hence views are inconsistent!

View inconsistency algorithm

Completeness-counterexample

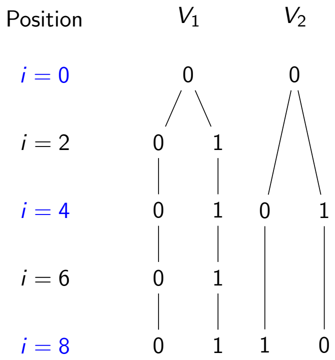
- Consider the views (nFOS) V_1 and V_2 with $T_1 = 2$ and $T_2 = 4$.

Behavior trees



- Consider the views (nFOS) V_1 and V_2 with $T_1 = 2$ and $T_2 = 4$.

Behavior trees



View inconsistency algorithm

Completeness-counterexample

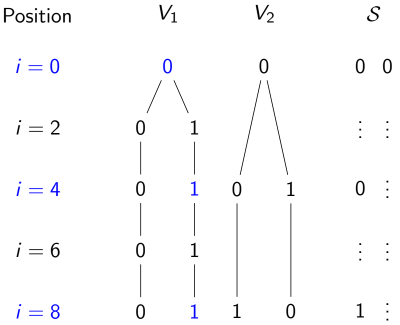
- Consider the views (nFOS) V_1 and V_2 with $T_1 = 2$ and $T_2 = 4$.

Behavior trees

Position	V_1	V_2	$\mathcal{S}$
$i = 0$	0	0	0 0
$i = 2$	0 1		⋮ ⋮
$i = 4$	0 1	0 1	0 ⋮
$i = 6$	0 1		⋮ ⋮
$i = 8$	0 1	1 0	1 ⋮

- Consider the views (nFOS) V_1 and V_2 with $T_1 = 2$ and $T_2 = 4$.

Behavior trees



$\rightarrow$ The views are inconsistent but the algorithm does not detect it.