States and Events in KandISTI

A Retrospective
in honour of Bernhard Steffen

Maurice H. ter Beek
ISTI–CNR, Pisa, Italy

joint work with

Alessandro Fantechi
UNIFI

Stefania Gnesi
ISTI–CNR

Franco Mazzanti
ISTI–CNR

B-Day

Limassol, Cyprus
4 November 2018

A long time ago in a galaxy far, far away...

According to Stefania and Alessandro

- The story started many years ago, more or less in the years following the success of CTL model checking [CES83/86]
- The process algebra world started thinking about developing action-based logics and verification frameworks
  - μ-calculus
  - ACTL/ACTL*: Action-based CTL/CTL*
  - μ-ACTL: ACTL with a fixpoint operator
  - AMC: ACTL Model Checker (with Franco)
  - FMC: Full μ-calculus Model Checker (with Franco)
  - JACK: Just Another Concurrency (with Franco)
  - SAM: A Symbolic Model Checker for ACTL* (with Franco)
  - etc.
- The results of these efforts are at the basis of KandISTI

ISoLA'18: Franco's 2nd consecutive RERS participation using KandISTI

A long time ago in a galaxy far, far away...

According to Stefania and Alessandro

- The story started many years ago, more or less in the years following the success of CTL model checking [CES83/86]
- The process algebra world started thinking about developing action-based logics and verification frameworks
  - μ-calculus
  - ACTL/ACTL*: Action-based CTL/CTL*
  - μ-ACTL: ACTL with a fixpoint operator
  - AMC: ACTL Model Checker (with Franco)
  - FMC: Full μ-calculus Model Checker (with Franco)
  - JACK: Just Another Concurrency (with Franco)
  - SAM: A Symbolic Model Checker for ACTL* (with Franco)
  - etc.
- The results of these efforts are at the basis of KandISTI

ISoLA'18: Franco's 2nd consecutive RERS participation using KandISTI

A long time ago in a galaxy far, far away...

According to Stefania and Alessandro

- The story started many years ago, more or less in the years following the success of CTL model checking [CES83/86]
- The process algebra world started thinking about developing action-based logics and verification frameworks
  - μ-calculus
  - ACTL/ACTL*: Action-based CTL/CTL*
  - μ-ACTL: ACTL with a fixpoint operator
  - AMC: ACTL Model Checker (with Franco)
  - FMC: Full μ-calculus Model Checker (with Franco)
  - JACK: Just Another Concurrency (with Franco)
  - SAM: A Symbolic Model Checker for ACTL* (with Franco)
  - etc.
- The results of these efforts are at the basis of KandISTI

ISoLA'18: Franco's 2nd consecutive RERS participation using KandISTI

A long time ago in a galaxy far, far away...

According to Stefania and Alessandro

- The story started many years ago, more or less in the years following the success of CTL model checking [CES83/86]
- The process algebra world started thinking about developing action-based logics and verification frameworks
  - μ-calculus
  - ACTL/ACTL*: Action-based CTL/CTL*
  - μ-ACTL: ACTL with a fixpoint operator
  - AMC: ACTL Model Checker (with Franco)
  - FMC: Full μ-calculus Model Checker (with Franco)
  - JACK: Just Another Concurrency (with Franco)
  - SAM: A Symbolic Model Checker for ACTL* (with Franco)
  - etc.
- The results of these efforts are at the basis of KandISTI

ISoLA'18: Franco's 2nd consecutive RERS participation using KandISTI

Aarhus, 1989–90 ?

Aarhus, 1989–90 ?

This may be considered the first model-checking framework for process algebrae

Aarhus, 1989–90 ?

This may be considered the first model-checking framework for process algebrae

International Conference on Computer Aided Verification

CAV 1989: Automatic Verification Methods for Finite State Systems pp 24-37 | Cite as

The concurrency workbench

Authors | Authors and affiliations

Rance Cleaveland, Joachim Parrow, Bernhard Steffen

Process Algebras And Systems Of Communicating Processes

First Online: 01 June 2005

Part of the Lecture Notes in Computer Science book series (LNCS, volume 407)

Abstract

The Concurrency Workbench is an automated tool that caters for the analysis of networks of finite-state processes expressed in Milner's Calculus of Communicating Systems. Its key feature is its scope: a variety of different verification methods, including equivalence checking, preorder checking, and model checking, are supported for several different process semantics. One experience from our work is that a large number of interesting verification methods can be

Aarhus, 1989–90 ?

In that period, Stefania, Alessandro and later Franco started working with the action-based logic ACTL, introduced by De Nicola & Vaandrager at LICS'90

Aarhus, 1989–90 ?

In that period, Stefania, Alessandro and later Franco started working with the action-based logic ACTL, introduced by De Nicola & Vaandrager at LICS'90

International Conference on Computer Aided Verification

CAV 1991: Computer Aided Verification pp 37-47 | Cite as

An action based framework for verifying logical and behavioural properties of concurrent systems

Authors | Authors and affiliations

R. De Nicola, A. Fantechi, S. Gnesi, G. Ristori

Session 2: Model Checking

First Online: 29 May 2005

Part of the Lecture Notes in Computer Science book series (LNCS, volume 575)

Abstract

A system is described which supports proofs of both behavioural and logical properties of concurrent systems; these are specified by means of a process algebra and its associated logics. The logic is an action based version of the branching time logic CTL which we call ACTL; it is interpreted over transition labelled structures while CTL is interpreted over state labelled ones.

Aarhus, 1989–90 ?

The next paper in the CAV'91 proceedings hints that this was perhaps the first time Stefania, Alessandro and Bernhard became aware of each other's work..?

Aarhus, 1989–90 ?

The next paper in the CAV'91 proceedings hints that this was perhaps the first time Stefania, Alessandro and Bernhard became aware of each other's work..?

International Conference on Computer Aided Verification

CAV 1991: Computer Aided Verification pp 48-58 | Cite as

A linear-time model-checking algorithm for the alternation-free modal mu-calculus

Authors   Authors and affiliations

Ranee Cleaveland, Bernhard Steffen

Session 2: Model Checking

First Online: 29 May 2005

Part of the Lecture Notes in Computer Science book series (LNCS, volume 575)

Abstract

We develop a model-checking algorithm for a logic that permits propositions to be defined with greatest and least fixed points of mutually recursive systems of equations. This logic is as expressive as the alternation-free fragment of the modal mu-calculus identified by Emerson and Lei, and it may therefore be used to encode a number of temporal logics and behavioral

Leiden, 1990–96: Maurice studied Computer Science

2007 I first became aware of Bernhard when Tiziana pointed me to 'his' Kripke
2011 Transition Systems (KTS) during FMICS'07/SCP'11 work using Doubly-Labelled Transition Systems (L^2TS)

Leiden, 1990–96: Maurice studied Computer Science

2007 I first became aware of Bernhard when Tiziana pointed me to ‘his’ Kripke
2011 Transition Systems (KTS) during FMICS’07/*SCP*’11 work using Doubly-Labelled Transition Systems (L^2TS)

2012 Followed by my involvement in ISoLA: 3rd consecutive participation as
2018 track organiser (great networking opportunities! work with Rolf, Axel, etc.)

2016 Edited (with Stefania) 2 special issues in *STTT* on FMICS/AVoCS’16 and
2018 Formal Methods in Transport Systems (agreed with Bernhard at ISoLA’16)

2016 Edited (with Axel, agreed at ISoLA’16) a section in *FoMaC* on a hot topic:
2018 QSPL (IEEE *TSE*’18 / FM’18): this is about to appear, right Bernhard?

2018 Given Bernhard’s fundamental roles in ISoLA, *STTT* and *FoMaC*, I sure
20?? hope our roads will cross many more times...

Leiden, 1990–96: Maurice studied Computer Science

2007 I first became aware of Bernhard when Tiziana pointed me to 'his' Kripke
2011 Transition Systems (KTS) during FMICS'07/*SCP*'11 work using Doubly-Labelled Transition Systems (L^2TS): I often cite his SAS'99 paper since ☺

2012 Followed by my involvement in ISoLA: 3rd consecutive participation as
2018 track organiser (great networking opportunities! work with Rolf, Axel, etc.)

2016 Edited (with Stefania) 2 special issues in *STTT* on FMICS/AVoCS'16 and
2018 Formal Methods in Transport Systems (agreed with Bernhard at ISoLA'16)

2016 Edited (with Axel, agreed at ISoLA'16) a section in *FoMaC* on a hot topic:
2018 QSPL (IEEE *TSE*'18 / FM'18): this is about to appear, right Bernhard?

2018 Given Bernhard's fundamental roles in ISoLA, *STTT* and *FoMaC*, I sure
20?? hope our roads will cross many more times...

M. Müller-Olm, D.A. Schmidt, B. Steffen, Model-checking: a tutorial introduction, SAS'99

Leiden, 1990–96: Maurice studied Computer Science

2007 I first became aware of Bernhard when Tiziana pointed me to 'his' Kripke
2011 Transition Systems (KTS) during FMICS'07/*SCP*'11 work using Doubly-Labelled Transition Systems (L^2TS): I often cite his SAS'99 paper since ☺

2012 Followed by my involvement in ISoLA: 3rd consecutive participation as
2018 track organiser (great networking opportunities! work with Rolf, Axel, etc.)

2016 Edited (with Stefania) 2 special issues in *STTT* on FMICS/AVoCS'16 and
2018 Formal Methods in Transport Systems (agreed with Bernhard at ISoLA'16)

2016 Edited (with Axel, agreed at ISoLA'16) a section in *FoMaC* on a hot topic:
2018 QSPL (IEEE *TSE*'18 / FM'18): this is about to appear, right Bernhard?

2018 Given Bernhard's fundamental roles in ISoLA, *STTT* and *FoMaC*, I sure
20?? hope our roads will cross many more times...

M. Müller-Olm, D.A. Schmidt, B. Steffen, Model-checking: a tutorial introduction, SAS'99

Leiden, 1990–96: Maurice studied Computer Science

2007 I first became aware of Bernhard when Tiziana pointed me to 'his' Kripke
2011 Transition Systems (KTS) during FMICS'07/*SCP*'11 work using Doubly-Labelled Transition Systems (L^2TS): I often cite his SAS'99 paper since ☺

2012 Followed by my involvement in ISoLA: 3rd consecutive participation as
2018 track organiser (great networking opportunities! work with Rolf, Axel, etc.)

2016 Edited (with Stefania) 2 special issues in *STTT* on FMICS/AVoCS'16 and
2018 Formal Methods in Transport Systems (agreed with Bernhard at ISoLA'16)

2016 Edited (with Axel, agreed at ISoLA'16) a section in *FoMaC* on a hot topic:
2018 QSPL (IEEE *TSE*'18 / FM'18): this is about to appear, right Bernhard?

2018 Given Bernhard's fundamental roles in ISoLA, *STTT* and *FoMaC*, I sure
20?? hope our roads will cross many more times...

M. Müller-Olm, D.A. Schmidt, B. Steffen, Model-checking: a tutorial introduction, SAS'99

Leiden, 1990–96: Maurice studied Computer Science

2007 I first became aware of Bernhard when Tiziana pointed me to 'his' Kripke
2011 Transition Systems (KTS) during FMICS'07/*SCP*'11 work using Doubly-Labelled Transition Systems (L^2TS): I often cite his SAS'99 paper since ☺

2012 Followed by my involvement in ISoLA: 3rd consecutive participation as
2018 track organiser (great networking opportunities! work with Rolf, Axel, etc.)

2016 Edited (with Stefania) 2 special issues in *STTT* on FMICS/AVoCS'16 and
2018 Formal Methods in Transport Systems (agreed with Bernhard at ISoLA'16)

2016 Edited (with Axel, agreed at ISoLA'16) a section in *FoMaC* on a hot topic:
2018 QSPL (IEEE *TSE*'18 / FM'18): this is about to appear, right Bernhard?

2018 Given Bernhard's fundamental roles in ISoLA, *STTT* and *FoMaC*, I sure
20?? hope our roads will cross many more times...

M. Müller-Olm, D.A. Schmidt, B. Steffen, Model-checking: a tutorial introduction, SAS'99

Leiden, 1990–96: Maurice studied Computer Science

2007 I first became aware of Bernhard when Tiziana pointed me to 'his' Kripke
2011 Transition Systems (KTS) during FMICS'07/*SCP*'11 work using Doubly-
Labelled Transition Systems (L^2TS): I often cite his SAS'99 paper since ☺

2012 Followed by my involvement in ISoLA: 3rd consecutive participation as
2018 track organiser (great networking opportunities! work with Rolf, Axel, etc.)

2016 Edited (with Stefania) 2 special issues in *STTT* on FMICS/AVoCS'16 and
2018 Formal Methods in Transport Systems (agreed with Bernhard at ISoLA'16)

2016 Edited (with Axel, agreed at ISoLA'16) a section in *FoMaC* on a hot topic:
2018 QSPL (IEEE *TSE*'18 / FM'18): this is about to appear, right Bernhard?

2018 Given Bernhard's fundamental roles in ISoLA, *STTT* and *FoMaC*, I sure
20?? hope our roads will cross many more times...

M. Müller-Olm, D.A. Schmidt, B. Steffen, Model-checking: a tutorial introduction, SAS'99

Our Festschrift contribution

Brief overview on models and logics dealing with states and events

- KTS, L^2TS, but also state/event systems of Graf & Loiseaux (TAPSOFT'93)
- μ-calculus, ACTL, but also SE-LTL and ARCTL

...followed by specificities of our KandISTI model checker and logic

http://fmt.isti.cnr.it/kandisti

Family of model checkers developed at ISTI–CNR for over 2 decades

- FMC (PDPTA'99), UMC (*SCP*'11), CMC (ACM *TOSEM*'12), VMC (FM'12)

Explicit-state on-the-fly model checking of properties in state-based and event-based branching-time temporal logics, building on (A)CTL

- e.g. UCTL (FMICS'07), SocL (FASE'08), v-ACTL (*JLAMP*'16)

Complexity is linear w.r.t. size of the model and size of the formula

---

R. Cleaveland, B. Steffen, A Linear-Time Model-Checking Algorithm for the Alternation-Free Modal Mu-Calculus. FMSD'93

Our Festschrift contribution

Brief overview on models and logics dealing with states and events

- KTS, L^2TS, but also state/event systems of Graf & Loiseaux (TAPSOFT'93)
- μ-calculus, ACTL, but also SE-LTL and ARCTL

...followed by specificities of our KandISTI model checker and logic

http://fmt.isti.cnr.it/kandisti

Family of model checkers developed at ISTI–CNR for over 2 decades

- FMC (PDPTA'99), UMC (*SCP*'11), CMC (ACM *TOSEM*'12), VMC (FM'12)

Explicit-state on-the-fly model checking of properties in state-based and event-based branching-time temporal logics, building on (A)CTL

- e.g. UCTL (FMICS'07), SocL (FASE'08), v-ACTL (*JLAMP*'16)

Complexity is linear w.r.t. size of the model and size of the formula

R. Cleaveland, B. Steffen, A Linear-Time Model-Checking Algorithm for the Alternation-Free Modal Mu-Calculus. FMSD'93

Our Festschrift contribution

Brief overview on models and logics dealing with states and events

- KTS, L^{2}TS, but also state/event systems of Graf & Loiseaux (TAPSOFT'93)
- μ-calculus, ACTL, but also SE-LTL and ARCTL

...followed by specificities of our KandISTI model checker and logic

http://fmt.isti.cnr.it/kandisti

Family of model checkers developed at ISTI–CNR for over 2 decades

FMC (PDPTA'99), UMC (*SCP*'11), CMC (ACM *TOSEM*'12), VMC (FM'12)

Explicit-state on-the-fly model checking of properties in state-based and event-based branching-time temporal logics, building on (A)CTL

e.g. UCTL (FMICS'07), SocL (FASE'08), v-ACTL (*JLAMP*'16)

Complexity is linear w.r.t. size of the model and size of the formula

---

R. Cleaveland, B. Steffen, A Linear-Time Model-Checking Algorithm for the Alternation-Free Modal Mu-Calculus. FMSD'93

Our Festschrift contribution

Brief overview on models and logics dealing with states and events

- KTS, L^2TS, but also state/event systems of Graf & Loiseaux (TAPSOFT'93)
- μ-calculus, ACTL, but also SE-LTL and ARCTL

...followed by specificities of our KandISTI model checker and logic

http://fmt.isti.cnr.it/kandisti

Family of model checkers developed at ISTI–CNR for over 2 decades

FMC (PDPTA'99), UMC (*SCP*'11), CMC (ACM *TOSEM*'12), VMC (FM'12)

Explicit-state on-the-fly model checking of properties in state-based and event-based branching-time temporal logics, building on (A)CTL

e.g. UCTL (FMICS'07), SocL (FASE'08), v-ACTL (*JLAMP*'16)

Complexity is linear w.r.t. size of the model and size of the formula

R. Cleaveland, B. Steffen, A Linear-Time Model-Checking Algorithm for the Alternation-Free Modal Mu-Calculus. FMSD'93

Our Festschrift contribution

Brief overview on models and logics dealing with states and events

- KTS, L^2TS, but also state/event systems of Graf & Loiseaux (TAPSOFT'93)
- μ-calculus, ACTL, but also SE-LTL and ARCTL

...followed by specificities of our KandISTI model checker and logic

`http://fmt.isti.cnr.it/kandisti`

Family of model checkers developed at ISTI–CNR for over 2 decades

FMC (PDPTA'99), UMC (*SCP*'11), CMC (ACM *TOSEM*'12), VMC (FM'12)

Explicit-state on-the-fly model checking of properties in state-based and event-based branching-time temporal logics, building on (A)CTL

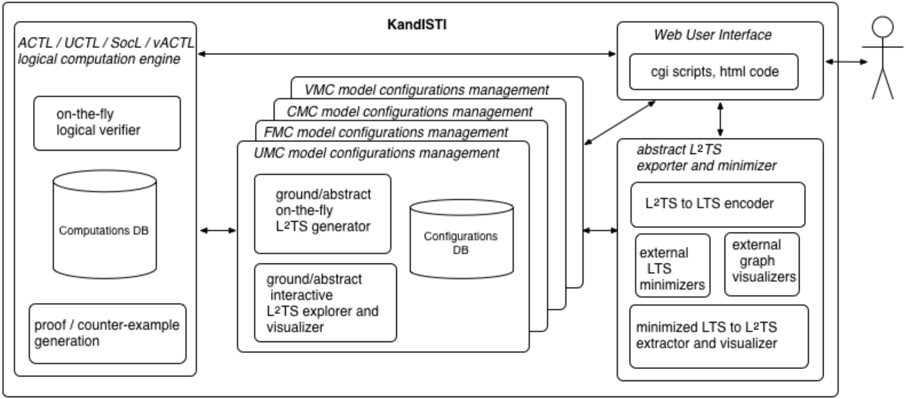
e.g. UCTL (FMICS'07), SocL (FASE'08), v-ACTL (*JLAMP*'16)

Complexity is linear w.r.t. size of the model and size of the formula

---

R. Cleaveland, B. Steffen, A Linear-Time Model-Checking Algorithm for the Alternation-Free Modal Mu-Calculus. FMSD'93

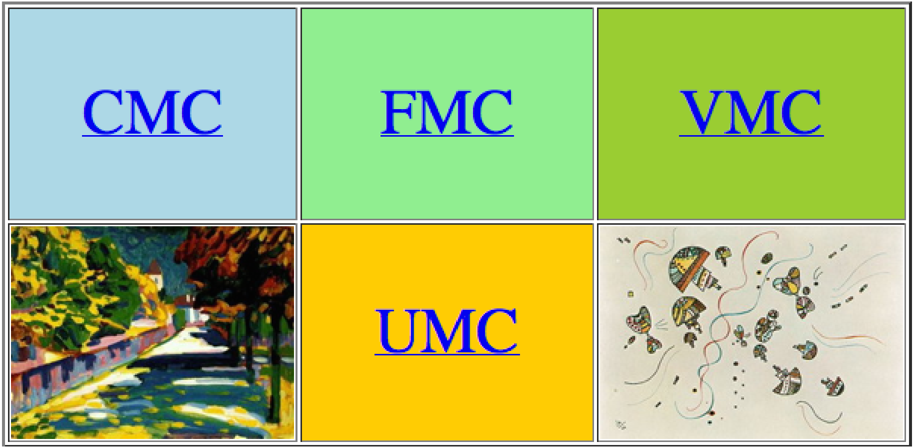
KandISTI architecture



Logical verification engine shared by all tools observes the underlying model as an abstract L^2TS independent from the operational semantics of the tool's specification language, thanks to an associated set of abstraction rules

M.H. ter Beek, S. Gnesi, F. Mazzanti, From EU Projects to a Family of Model Checkers: From Kandinsky to KandISTI, Festschrift Martin Wirsing, 2015

KandISTI screenshot



Explicit abstraction mechanism allows to specify which details of the model become observable labels on the L^{2}TS' states and transitions

M.H. ter Beek, S. Gnesi, F. Mazzanti, From EU Projects to a Family of Model Checkers: From Kandinsky to KandISTI, Festschrift Martin Wirsing, 2015

KandISTI logic (L^{2}TS semantics)

- Parametric state predicates (state labels)
  e.g. `pred1(arg1,arg2)`, `pred2`, `pred3(*,arg3)`
- Parametric event formulae (Bool expressions over transition labels)
  e.g. `(act1(arg1,arg2) or act2)`, `not act3(arg3,*,*)`
- Box, Diamond, fixpoint operators (i.e. full modal μ-calculus)
  e.g. `max` Y: `max` Z: (($\langle$`act2(arg1)`$\rangle$ Y) `or` $\langle$`act2`$\rangle$ Z)
- CTL operators (e.g. neXt, Always, (Weak) Until, Globally, Eventually)
  e.g. *EX* `pred1`, *A*[`pred1(arg1)` *U* `pred2`], *AG EF* `pred1`
- ACTL-like operators (i.e. event-based variants of CTL operators)
  e.g. *EX* {`act1`} *true*, *A*[`pred1(arg1)` {`act1`} *U* {`act2`} `pred2`]
- Parametric formulae that express data correlations
  e.g. [`act1`($1, $2)] *AF* {`act2`(%1,%2)} *true*, *EF* {$1} *EF* {%1} *true*
- Deontic variants of some of the above operators (for MTS, next slide)
  e.g. $\langle$`act1`$\rangle$# *true*, *EF*# {`act`} `pred1`
- Special-purpose predefined state predicates
  e.g. `PRINT(msg,arg1,arg2)`, `DEPTH_LT_n`, `FINAL`

KandISTI logic (L^{2}TS semantics)

- Parametric state predicates (state labels)
  e.g. $\texttt{pred1}(\texttt{arg1}, \texttt{arg2})$, $\texttt{pred2}$, $\texttt{pred3}(*, \texttt{arg3})$
- Parametric event formulae (Bool expressions over transition labels)
  e.g. $(\texttt{act1}(\texttt{arg1}, \texttt{arg2})\ \texttt{or}\ \texttt{act2})$, $\texttt{not}\ \texttt{act3}(\texttt{arg3}, *, *)$
- Box, Diamond, fixpoint operators (i.e. full modal μ-calculus)
  e.g. $\texttt{max}\ Y\!: \texttt{max}\ Z\!: ((\langle \texttt{act2}(\texttt{arg1}) \rangle\, Y)\ \texttt{or}\ \langle \texttt{act2} \rangle\, Z)$
- CTL operators (e.g. neXt, Always, (Weak) Until, Globally, Eventually)
  e.g. $EX\ \texttt{pred1}$, $A[\texttt{pred1}(\texttt{arg1})\ U\ \texttt{pred2}]$, $AG\ EF\ \texttt{pred1}$
- ACTL-like operators (i.e. event-based variants of CTL operators)
  e.g. $EX\,\{\texttt{act1}\}\,\mathit{true}$, $A[\texttt{pred1}(\texttt{arg1})\ \{\texttt{act1}\}\,U\,\{\texttt{act2}\}\ \texttt{pred2}]$
- Parametric formulae that express data correlations
  e.g. $[\texttt{act1}(\$1, \$2)]\, AF\, \{\texttt{act2}(\%1, \%2)\}\, \mathit{true}$, $EF\, \{\$1\}\, EF\, \{\%1\}\, \mathit{true}$
- Deontic variants of some of the above operators (for MTS, next slide)
  e.g. $\langle \texttt{act1} \rangle \#\ \mathit{true}$, $EF\#\,\{\texttt{act}\}\ \texttt{pred1}$
- Special-purpose predefined state predicates
  e.g. $\texttt{PRINT}(\texttt{msg}, \texttt{arg1}, \texttt{arg2})$, $\texttt{DEPTH\_LT\_n}$, $\texttt{FINAL}$

KandISTI logic (L^{2}TS semantics)

- Parametric state predicates (state labels)
  e.g. `pred1(arg1,arg2)`, `pred2`, `pred3(*,arg3)`
- Parametric event formulae (Bool expressions over transition labels)
  e.g. `(act1(arg1,arg2) or act2)`, `not act3(arg3,*,*)`
- Box, Diamond, fixpoint operators (i.e. full modal μ-calculus)
  e.g. `max` Y: `max` Z: $((\langle$`act2(arg1)`$\rangle\, Y)$ `or` $\langle$`act2`$\rangle\, Z)$
- CTL operators (e.g. neXt, Always, (Weak) Until, Globally, Eventually)
  e.g. EX `pred1`, A[`pred1(arg1)` U `pred2`], $AG\ EF$ `pred1`
- ACTL-like operators (i.e. event-based variants of CTL operators)
  e.g. EX {`act1`} *true*, A[`pred1(arg1)` {`act1`} U {`act2`} `pred2`]
- Parametric formulae that express data correlations
  e.g. [`act1`($1,$2)] AF {`act2`(%1,%2)} *true*, EF {$1} EF {%1} *true*
- Deontic variants of some of the above operators (for MTS, next slide)
  e.g. $\langle$`act1`$\rangle$# *true*, EF# {`act`} `pred1`
- Special-purpose predefined state predicates
  e.g. `PRINT(msg,arg1,arg2)`, `DEPTH_LT_n`, `FINAL`

KandISTI logic (L^{2}TS semantics)

- Parametric state predicates (state labels)
  e.g. `pred1`$($`arg1`$,$`arg2`$)$, `pred2`, `pred3`$(*,$`arg3`$)$
- Parametric event formulae (Bool expressions over transition labels)
  e.g. $($`act1`$($`arg1`$,$`arg2`$)$ `or` `act2`$)$, `not` `act3`$($`arg3`$,*,*)$
- Box, Diamond, fixpoint operators (i.e. full modal μ-calculus)
  e.g. `max` Y: `max` Z: $((\langle$`act2`$($`arg1`$)\rangle\ Y)$ `or` $\langle$`act2`$\rangle\ Z)$
- CTL operators (e.g. neXt, Always, (Weak) Until, Globally, Eventually)
  e.g. *EX* `pred1`, *A*[`pred1`(`arg1`) *U* `pred2`], *AG EF* `pred1`
- ACTL-like operators (i.e. event-based variants of CTL operators)
  e.g. *EX* {`act1`} *true*, *A*[`pred1`(`arg1`) {`act1`} *U* {`act2`} `pred2`]
- Parametric formulae that express data correlations
  e.g. [`act1`($1, $2)] *AF* {`act2`(%1,%2)} *true*, *EF* {$1} *EF* {%1} *true*
- Deontic variants of some of the above operators (for MTS, next slide)
  e.g. ⟨`act1`⟩# *true*, *EF*# {`act`} `pred1`
- Special-purpose predefined state predicates
  e.g. `PRINT(msg, arg1, arg2)`, `DEPTH_LT_n`, `FINAL`

KandISTI logic (L^{2}TS semantics)

- Parametric state predicates (state labels)
  e.g. `pred1`(`arg1`, `arg2`), `pred2`, `pred3`($*$, `arg3`)
- Parametric event formulae (Bool expressions over transition labels)
  e.g. (`act1`(`arg1`, `arg2`) `or` `act2`), `not act3`(`arg3`, $*$, $*$)
- Box, Diamond, fixpoint operators (i.e. full modal μ-calculus)
  e.g. `max` Y: `max` Z: (($\langle$`act2`(`arg1`)$\rangle$ Y) `or` $\langle$`act2`$\rangle$ Z)
- CTL operators (e.g. neXt, Always, (Weak) Until, Globally, Eventually)
  e.g. *EX* `pred1`, *A* [`pred1`(`arg1`) *U* `pred2`], *AG EF* `pred1`
- ACTL-like operators (i.e. event-based variants of CTL operators)
  e.g. *EX* {`act1`} *true*, *A* [`pred1`(`arg1`) {`act1`} *U* {`act2`} `pred2`]
- Parametric formulae that express data correlations
  e.g. [`act1`($1, $2)] *AF* {`act2`(%1,%2)} *true*, *EF* {$1} *EF* {%1} *true*
- Deontic variants of some of the above operators (for MTS, next slide)
  e.g. $\langle$`act1`$\rangle$# *true*, *EF*# {`act`} `pred1`
- Special-purpose predefined state predicates
  e.g. `PRINT`(`msg`, `arg1`, `arg2`), `DEPTH_LT_n`, `FINAL`

KandISTI logic (L^2TS semantics)

- Parametric state predicates (state labels)
  e.g. `pred1`$($`arg1`$,$ `arg2`$)$, `pred2`, `pred3`$(*,$ `arg3`$)$
- Parametric event formulae (Bool expressions over transition labels)
  e.g. $($`act1`$($`arg1`$,$`arg2`$)$ `or` `act2`$)$, `not` `act3`$($`arg3`$, *, *)$
- Box, Diamond, fixpoint operators (i.e. full modal μ-calculus)
  e.g. `max` Y: `max` Z: $((\langle$`act2`$($`arg1`$)\rangle\ Y)$ `or` $\langle$`act2`$\rangle\ Z)$
- CTL operators (e.g. neXt, Always, (Weak) Until, Globally, Eventually)
  e.g. EX `pred1`, $A[$`pred1`$($`arg1`$)\ U$ `pred2`$]$, $AG\ EF$ `pred1`
- ACTL-like operators (i.e. event-based variants of CTL operators)
  e.g. $EX\ \{$`act1`$\}$ *true*, $A[$`pred1`$($`arg1`$)\ \{$`act1`$\}\ U\ \{$`act2`$\}$ `pred2`$]$
- Parametric formulae that express data correlations
  e.g. $[$`act1`$(\$1, \$2)]\ AF\ \{$`act2`$(\%1, \%2)\}$ *true*, $EF\ \{\$1\}\ EF\ \{\%1\}$ *true*
- Deontic variants of some of the above operators (for MTS, next slide)
  e.g. $\langle$`act1`$\rangle\#$ *true*, $EF\#\{$`act`$\}$ `pred1`
- Special-purpose predefined state predicates
  e.g. `PRINT`$($`msg`$,$ `arg1`$,$ `arg2`$)$, `DEPTH_LT_n`, `FINAL`

KandISTI logic (L^2TS semantics)

- Parametric state predicates (state labels)
  e.g. `pred1(arg1, arg2)`, `pred2`, `pred3(*, arg3)`
- Parametric event formulae (Bool expressions over transition labels)
  e.g. `(act1(arg1, arg2) or act2)`, `not act3(arg3, *, *)`
- Box, Diamond, fixpoint operators (i.e. full modal μ-calculus)
  e.g. `max` Y: `max` Z : (($\langle$`act2(arg1)`$\rangle$ Y) `or` $\langle$`act2`$\rangle$ Z)
- CTL operators (e.g. neXt, Always, (Weak) Until, Globally, Eventually)
  e.g. EX `pred1`, A[`pred1(arg1)` U `pred2`], $AG\ EF$ `pred1`
- ACTL-like operators (i.e. event-based variants of CTL operators)
  e.g. EX {`act1`} *true*, A[`pred1(arg1)` {`act1`} U {`act2`} `pred2`]
- Parametric formulae that express data correlations
  e.g. [`act1`(\$1, \$2)] AF {`act2`(%1,%2)} *true*, EF {\$1} EF {%1} *true*
- Deontic variants of some of the above operators (for MTS, next slide)
  e.g. $\langle$`act1`$\rangle$# *true*, EF# {`act`} `pred1`
- Special-purpose predefined state predicates
  e.g. `PRINT(msg, arg1, arg2)`, `DEPTH_LT_n`, `FINAL`

KandISTI logic (L^{2}TS semantics)

- Parametric state predicates (state labels)
  e.g. `pred1(arg1,arg2)`, `pred2`, `pred3(*,arg3)`
- Parametric event formulae (Bool expressions over transition labels)
  e.g. `(act1(arg1,arg2) or act2)`, `not act3(arg3,*,*)`
- Box, Diamond, fixpoint operators (i.e. full modal μ-calculus)
  e.g. `max` Y: `max` Z: $((\langle$`act2(arg1)`$\rangle\, Y)$ `or` $\langle$`act2`$\rangle\, Z)$
- CTL operators (e.g. neXt, Always, (Weak) Until, Globally, Eventually)
  e.g. *EX* `pred1`, *A* [`pred1(arg1)` *U* `pred2`], *AG EF* `pred1`
- ACTL-like operators (i.e. event-based variants of CTL operators)
  e.g. *EX* {`act1`} *true*, *A* [`pred1(arg1)` {`act1`} *U* {`act2`} `pred2`]
- Parametric formulae that express data correlations
  e.g. [`act1`($1, $2)] *AF* {`act2`(%1,%2)} *true*, *EF* {$1} *EF* {%1} *true*
- Deontic variants of some of the above operators (for MTS, next slide)
  e.g. $\langle$`act1`$\rangle$# *true*, *EF*# {`act`} `pred1`
- Special-purpose predefined state predicates
  e.g. `PRINT(msg,arg1,arg2)`, `DEPTH_LT_n`, `FINAL`

KandISTI logic (L^2TS semantics)

- Parametric state predicates (state labels)
  e.g. `pred1(arg1,arg2)`, `pred2`, `pred3(*,arg3)`
- Parametric event formulae (Bool expressions over transition labels)
  e.g. `(act1(arg1,arg2) or act2)`, `not act3(arg3,*,*)`
- Box, Diamond, fixpoint operators (i.e. full modal μ-calculus)
  e.g. `max` *Y*: `max` *Z*: ((⟨`act2(arg1)`⟩ *Y*) `or` ⟨`act2`⟩ *Z*)
- CTL operators (e.g. neXt, Always, (Weak) Until, Globally, Eventually)
  e.g. *EX* `pred1`, *A*[`pred1(arg1)` *U* `pred2`], *AG EF* `pred1`
- ACTL-like operators (i.e. event-based variants of CTL operators)
  e.g. *EX* {`act1`} *true*, *A*[`pred1(arg1)` {`act1`} *U* {`act2`} `pred2`]
- Parametric formulae that express data correlations
  e.g. [`act1`(\$1, \$2)] *AF* {`act2`(%1,%2)} *true*, *EF* {\$1} *EF* {%1} *true*
- Deontic variants of some of the above operators (for MTS, next slide)
  e.g. ⟨`act1`⟩# *true*, *EF*# {`act`} `pred1`
- Special-purpose predefined state predicates
  e.g. `PRINT(msg,arg1,arg2)`, `DEPTH_LT_n`, `FINAL`

Application: Software Product Lines

Modal Transition Systems (MTS) by Larsen & Thomsen at LICS'88

- LTS distinguishing possible (may) and required (must) transitions

MTS with variability constraints (MTSυ) by the four of us in *JLAMP*'16

- additional variability constraints (mimicking feature models) to be able to decide which implementations (LTS) are product variants

Application: Software Product Lines

Modal Transition Systems (MTS) by Larsen & Thomsen at LICS'88

- LTS distinguishing possible (may) and required (must) transitions

MTS with variability constraints (MTSυ) by the four of us in *JLAMP*'16

- additional variability constraints (mimicking feature models) to be able to decide which implementations (LTS) are product variants

Application: Software Product Lines

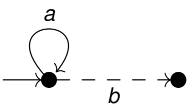
Modal Transition Systems (MTS) by Larsen & Thomsen at LICS'88

- LTS distinguishing possible (may) and required (must) transitions

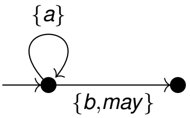
MTS with variability constraints (MTSυ) by the four of us in *JLAMP*'16

- additional variability constraints (mimicking feature models) to be able to decide which implementations (LTS) are product variants

MTS + v-ACTL can be interpreted as L^{2}TS + ACTL:



$$T = a.T + b(may).nil$$



$\langle a \rangle \# \; true$ $\qquad\qquad$ $\langle a \wedge not\ may \rangle \; true$

From a v-ACTL formula and an MTS representation (left) of a process (middle) to a corresponding ACTL formula over the L^{2}TS interpretation (right) of the MTS

Application: Software Product Lines

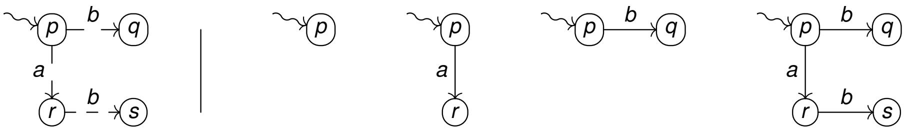
Modal Transition Systems (MTS) by Larsen & Thomsen at LICS'88

- LTS distinguishing possible (may) and required (must) transitions

MTS with variability constraints (MTSυ) by the four of us in *JLAMP*'16

- additional variability constraints (mimicking feature models) to be able to decide which implementations (LTS) are product variants

An MTS and four implementation variants (LTS):



With variability constraint *a* ALT *b*, only the two central LTS are product variants

Application: Software Product Lines

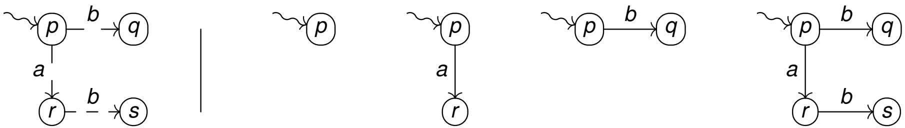
Modal Transition Systems (MTS) by Larsen & Thomsen at LICS'88

- LTS distinguishing possible (may) and required (must) transitions

MTS with variability constraints (MTSυ) by the four of us in *JLAMP*'16

- additional variability constraints (mimicking feature models) to be able to decide which implementations (LTS) are product variants

An MTS and four implementation variants (LTS):



With variability constraint *a* ALT *b*, only the two central LTS are product variants

In *SCP*'19 we prove MTSυ equally expressive as Featured Transition Systems (FTS) introduced by Axel et al. at ICSE'10 and IEEE *TSE*'13

Aim: lift known formal methods and tools from single system (product) to a set of products (family) by exploiting variability modelling and analysis

VMC offers both product-based and family-based variability analyses

1. The set of all product variants can explicitly be generated and the resulting LTS verified against a logic property (expressed in ACTL)
2. A logic property (expressed in v-ACTL) can be verified on the MTS, relying on the fact that under certain conditions its validity implies validity of the same property for all its product variants (next slide)

VMC: Variability Model Checker

Aim: lift known formal methods and tools from single system (product) to a set of products (family) by exploiting variability modelling and analysis

VMC offers both product-based and family-based variability analyses

1. The set of all product variants can explicitly be generated and the resulting LTS verified against a logic property (expressed in ACTL)
2. A logic property (expressed in v-ACTL) can be verified on the MTS, relying on the fact that under certain conditions its validity implies validity of the same property for all its product variants (next slide)

VMC: Variability Model Checker

Aim: lift known formal methods and tools from single system (product) to a set of products (family) by exploiting variability modelling and analysis

VMC offers both product-based and family-based variability analyses

1. The set of all product variants can explicitly be generated and the resulting LTS verified against a logic property (expressed in ACTL)
2. A logic property (expressed in v-ACTL) can be verified on the MTS, relying on the fact that under certain conditions its validity implies validity of the same property for all its product variants (next slide)

VMC: Variability Model Checker

Aim: lift known formal methods and tools from single system (product) to a set of products (family) by exploiting variability modelling and analysis

VMC offers both product-based and family-based variability analyses

1. The set of all product variants can explicitly be generated and the resulting LTS verified against a logic property (expressed in ACTL)
2. A logic property (expressed in v-ACTL) can be verified on the MTS, relying on the fact that under certain conditions its validity implies validity of the same property for all its product variants (next slide)

Preservation of formulae in v-ACTL

v-ACTL$^\Box$/v-ACTLive$^\Box$:

$$\begin{aligned}\phi \ ::= \ & \mathit{false} \mid \mathit{true} \mid \phi \ \mathit{and}\ \phi \mid \phi \ \mathit{or}\ \phi \mid [\chi]\,\phi \mid \langle\chi\rangle\#\,\phi \mid \\ & \mathit{EX}\#\,\phi \mid \mathit{EX}\#\,\{\chi\}\,\phi \mid \mathit{EF}\#\,\phi \mid \mathit{EF}\#\,\{\chi\}\,\phi \mid \\ & \mathit{AF}\#\,\phi \mid \mathit{AF}\#\,\{\chi\}\,\phi \mid \mathit{AG}\,\phi \mid \mathit{AF}\,\phi \mid \mathit{AF}\,\{\chi\}\,\phi\end{aligned}$$

any formula that is true for MTSυ, is also true for all products (LTS)

v-ACTL$^-$:

$$\begin{aligned}\psi \ ::= \ & \mathit{false} \mid \mathit{true} \mid \psi \ \mathit{and}\ \psi \mid \psi \ \mathit{or}\ \psi \mid [\chi]\#\,\psi \mid \langle\chi\rangle\,\psi \mid \\ & \mathit{EX}\,\phi \mid \mathit{EX}\,\{\chi\}\,\phi \mid \mathit{EF}\,\psi \mid \mathit{EF}\{\chi\}\,\psi \mid \mathit{AF}\,\phi \mid \mathit{AF}\,\{\chi\}\,\phi\end{aligned}$$

any formula that is false for MTS, is also false for all products (LTS)

Preservation of formulae in v-ACTL

v-ACTL$^{\Box}$/v-ACTLive$^{\Box}$:

$$\begin{aligned}\phi ::= \;& \mathit{false} \mid \mathit{true} \mid \phi \;\mathit{and}\; \phi \mid \phi \;\mathit{or}\; \phi \mid [\chi]\,\phi \mid \langle\chi\rangle\#\,\phi \mid \\ & \mathit{EX}\#\,\phi \mid \mathit{EX}\#\,\{\chi\}\,\phi \mid \mathit{EF}\#\,\phi \mid \mathit{EF}\#\,\{\chi\}\,\phi \mid \\ & \mathit{AF}\#\,\phi \mid \mathit{AF}\#\,\{\chi\}\,\phi \mid \mathit{AG}\,\phi \mid \mathit{AF}\,\phi \mid \mathit{AF}\,\{\chi\}\,\phi\end{aligned}$$

any formula that is true for MTSυ, is also true for all products (LTS)

v-ACTL^{-}:

$$\begin{aligned}\psi ::= \;& \mathit{false} \mid \mathit{true} \mid \psi \;\mathit{and}\; \psi \mid \psi \;\mathit{or}\; \psi \mid [\chi]\#\,\psi \mid \langle\chi\rangle\,\psi \mid \\ & \mathit{EX}\,\phi \mid \mathit{EX}\,\{\chi\}\,\phi \mid \mathit{EF}\,\psi \mid \mathit{EF}\{\chi\}\,\psi \mid \mathit{AF}\,\phi \mid \mathit{AF}\,\{\chi\}\,\phi\end{aligned}$$

any formula that is false for MTS, is also false for all products (LTS)

Live states use SPL-specific information

$$S \models \phi \;\Rightarrow\; S_p \models \phi \;\; \forall \text{ product LTS } S_p \text{ of MTS}\upsilon\ S$$

Recall: all (reachable) must transitions are preserved in product LTS

Live action sets define live states (do not occur as final in any product)

MTSυ

Constraint

a OR b

LTS

In any product in which p occurs,
p has at least one outgoing transition

$\Rightarrow$ p is a live state, since a OR b gives rise to a live action set $\{a, b\}$

Live states use SPL-specific information

$$S \models \phi \;\Rightarrow\; S_p \models \phi \;\; \forall \text{product LTS } S_p \text{ of MTS}\upsilon\ S$$

Recall: all (reachable) must transitions are preserved in product LTS

Live action sets define live states (do not occur as final in any product)

MTSυ

Constraint

a OR b

LTS

In any product in which p occurs,
p has at least one outgoing transition

$\Rightarrow$ p is a live state, since a OR b gives rise to a live action set $\{a, b\}$

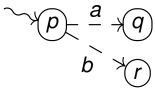
Live states use SPL-specific information

$$S \models \phi \;\Rightarrow\; S_p \models \phi \;\; \forall \text{ product LTS } S_p \text{ of MTS}\upsilon\ S$$

Recall: all (reachable) must transitions are preserved in product LTS

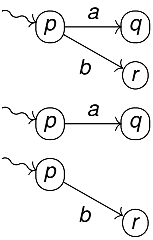
Live action sets define live states (do not occur as final in any product)

MTSυ



Constraint

a OR *b*



LTS

In any product in which *p* occurs,
p has at least one outgoing transition

$\Rightarrow$ *p* is a live state, since *a* OR *b* gives rise to a live action set $\{a, b\}$

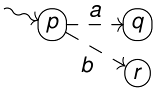
Live states use SPL-specific information

$$S \models \phi \;\Rightarrow\; S_p \models \phi \;\; \forall \text{ product LTS } S_p \text{ of MTS}\upsilon\ S$$

Recall: all (reachable) must transitions are preserved in product LTS

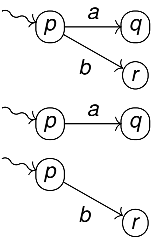
Live action sets define live states (do not occur as final in any product)

MTSυ



Constraint

a OR b



LTS

In any product in which p occurs,
p has at least one outgoing transition

$\Rightarrow$ p is a live state, since a OR b gives rise to a live action set $\{a, b\}$

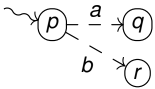
Live states use SPL-specific information

$$S \models \phi \;\Rightarrow\; S_p \models \phi \;\; \forall \text{product LTS } S_p \text{ of MTS}\upsilon\ S$$

Recall: all (reachable) must transitions are preserved in product LTS

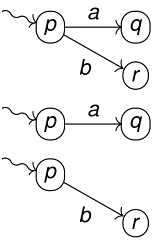
Live action sets define live states (do not occur as final in any product)

MTSυ



Constraint

a OR *b*



LTS

In any product in which *p* occurs,
p has at least one outgoing transition

$\Rightarrow$ *p* is a live state, since *a* OR *b* gives rise to a live action set $\{a, b\}$

Thanks!

From Pisa we wish Bernhard many more...

- papers
- STTT's
- ISoLA's
- FoMaC's
- etc.
- but above all ...
- many more birthdays!!!
- in good health
- with family and friends!

Thanks!

From Pisa we wish Bernhard many more...

- papers
- STTT's
- ISoLA's
- FoMaC's
- etc.
- but above all ...
- many more birthdays!!!
- in good health
- with family and friends!

Thanks!

From Pisa we wish Bernhard many more...

- papers
- STTT's
- ISoLA's
- FoMaC's
- etc.
- but above all ...
- many more birthdays!!!
- in good health
- with family and friends!

Thanks!

From Pisa we wish Bernhard many more...

- papers
- STTT's
- ISoLA's
- FoMaC's
- etc.
- but above all ...
- many more birthdays!!!
- in good health
- with family and friends!

Thanks!

From Pisa we wish Bernhard many more...

- papers
- STTT's
- ISoLA's
- FoMaC's
- etc.
- but above all ...
- many more birthdays!!!
- in good health
- with family and friends!

Thanks!

From Pisa we wish Bernhard many more...

- papers
- STTT's
- ISoLA's
- FoMaC's
- etc.
- but above all ...
- many more birthdays!!!
- in good health
- with family and friends!