Asynchronous Team Automata

Maurice ter Beek with Davide Basile and José Proença

FMT, CNR–ISTI, Pisa, Italy FMT, CNR–ISTI, Pisa, Italy CISTER & University of Porto, Portugal

APM@ETAPS 2026, Turin, Italy, April 17th, 2026

Outline

- 25+ Years of Team Automata
  - Motivation and development
  - Constrained multi-party synchronisations
- Asynchronous Team Automata
  - Inspiration and challenge
  - Syntax and semantics by example
  - Well-formedness and well-behavedness
  - Tool support: A-Team

25+ Years of Team Automata

Team Automata

A formalism for interacting component-based systems, whereby multiple sending and receiving actions from concurrent automata can synchronise on certain executions

First proposed at the 1997 ACM SIGGROUP Conference on Supporting Group Work for modelling components of groupware systems and their interconnections [GROUP'97]

Inspired by Input/Output (I/O) automata, inheriting the distinction between internal and external (input and output) actions used for communication with the environment

Team Automata

A formalism for interacting component-based systems, whereby multiple sending and receiving actions from concurrent automata can synchronise on certain executions

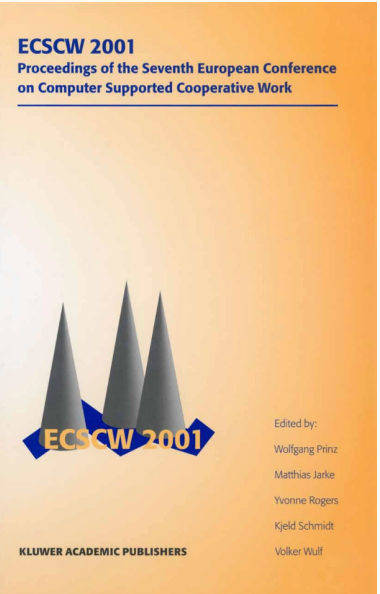
First proposed at the 1997 ACM SIGGROUP Conference on Supporting Group Work for modelling components of groupware systems and their interconnections [GROUP'97]

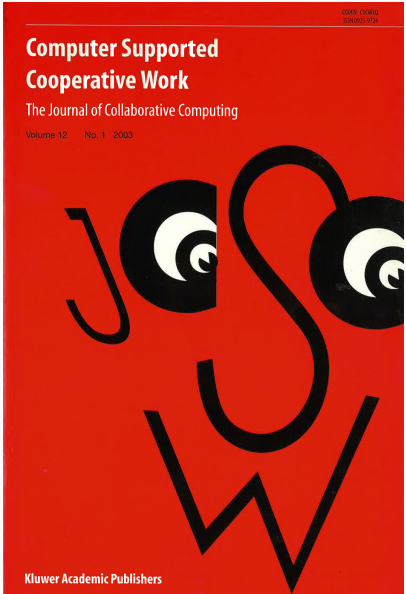
Inspired by Input/Output (I/O) automata, inheriting the distinction between internal and external (input and output) actions used for communication with the environment

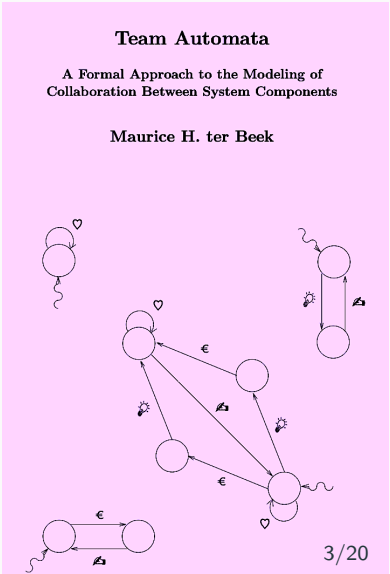
Formally defined in Computer Supported Cooperative Work (CSCW) — The Journal of Collaborative Computing, as composed by component automata that synchronise [CSCW'03]

Technically an extension of I/O automata, imposing hardly any restrictions on the role of actions in components, while composition is not limited to the synchronous product [IPL'05]

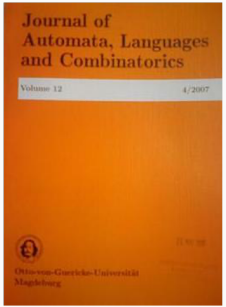
1997-2003: Team Automata and CSCW







post-2003: Team Automata and Formal Languages



Contents lists available at SciVerse ScienceDirect

Theoretical Computer Science

journal homepage: www.elsevier.com/locate/tcs

Vector team automata

Maurice H. ter Beek [a,*], Jetty Kleijn [b]

[a] ISTI–CNR, Via G. Moruzzi 1, 56124 Pisa, Italy

[b] LIACS, Universiteit Leiden, P.O. Box 9512, 2300 RA Leiden, The Netherlands

Fundamenta Informaticae 91 (2009) 437–461

DOI 10.3233/FI-2009-0051

IOS Press

TEAMS OF PUSHDOWN AUTOMATA

MAURICE H. TER BEEK[a,*], ERZSÉBET CSUHAJ-VARJÚ[b,†] and VICTOR MITRANA[c,‡]

[a]*Istituto di Scienza e Tecnologie dell'Informazione, Consiglio Nazionale delle Ricerche, Via G. Moruzzi 1, 56124 Pisa, Italy;*

[b]*Computer and Automation Research Institute, Hungarian Academy of Sciences, Kende utca 13-17, 1111 Budapest, Hungary;*

[c]*Faculty of Mathematics, University of Bucharest, Str. Academiei 14, 70109 Bucharest, Romania*

Available online at www.sciencedirect.com

ScienceDirect

Electronic Notes in Theoretical Computer Science 195 (2008) 41–55

Electronic Notes in Theoretical Computer Science

www.elsevier.com/locate/entcs

Associativity of Infinite Synchronized Shuffles and Team Automata

Maurice H. ter Beek*

Istituto di Scienza e Tecnologie dell'Informazione "A. Faedo", CNR

Via G. Moruzzi 1, 56124 Pisa, Italy

maurice.terbeek@isti.cnr.it

Jetty Kleijn†

Leiden Institute of Advanced Computer Science, Universiteit Leiden

P.O. Box 9512, 2300 RA Leiden, The Netherlands

kleijn@liacs.nl

A Calculus for Team Automata*

Maurice H. ter Beek[1]

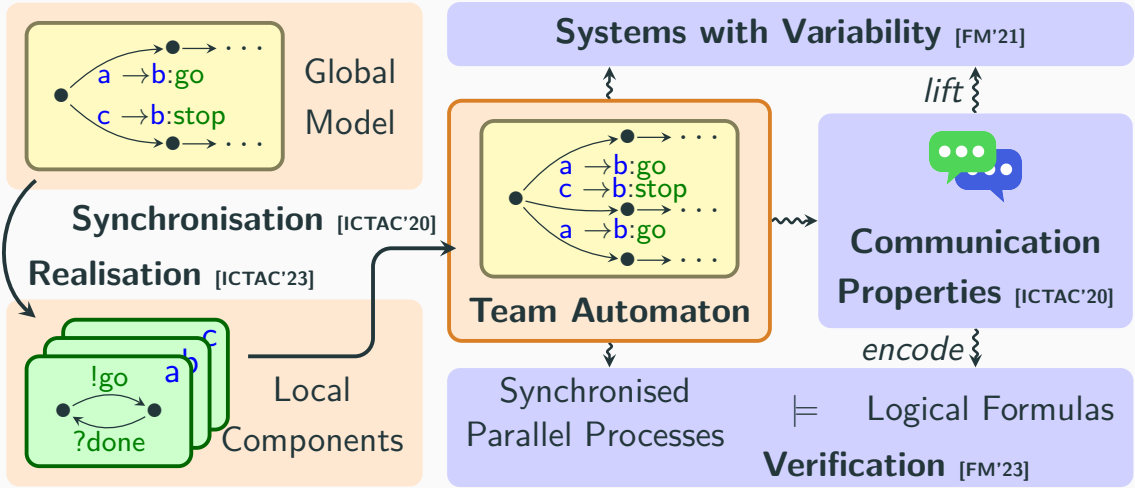
Istituto di Scienza e Tecnologie dell'Informazione, CNR via G. Moruzzi 1, 56124 Pisa, Italy

Fabio Gadducci[2]

Dipartimento di Informatica, Università di Pisa via Buonarroti 2, 56125 Pisa, Italy

Dirk Janssens[3]

post-2003: Team Automata and Formal Methods



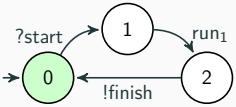
[FACS'23]: Overview on Constrained Multiparty Synchronisation in Team Automata

[COORDINATION'24]: Team Automata: Overview and Roadmap.

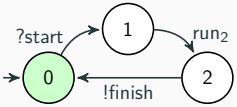
Constrained Multi-party Synchronisations

Team Automata by Example

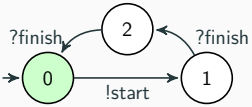
Team Automata: **not all system transitions are meaningful!**



$\mathcal{R}unner_1$



$\mathcal{R}unner_2$



Controller

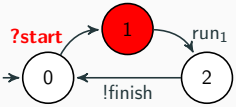
Team Automata

[CSCW'01'03] [FM'03,'21,'23] [TCS'12'13]

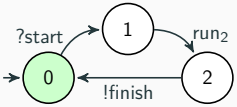
[COORDINATION'17,'20,'24] [ICTAC'20,'23]

Team Automata by Example

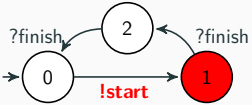
Team Automata: **not all system transitions are meaningful!**



$\mathcal{R}unner_1$



$\mathcal{R}unner_2$



Controller

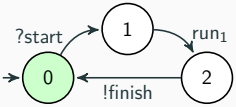
Team Automata

[CSCW'01'03] [FM'03,'21,'23] [TCS'12'13]

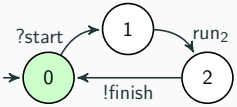
[COORDINATION'17,'20,'24] [ICTAC'20,'23]

Team Automata by Example

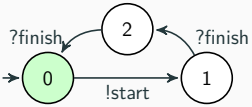
Team Automata: **not all system transitions are meaningful!**



$\mathcal{R}unner_1$



$\mathcal{R}unner_2$



Controller

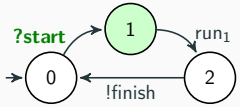
Team Automata

[CSCW'01'03] [FM'03,'21,'23] [TCS'12'13]

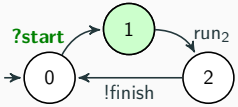
[COORDINATION'17,'20,'24] [ICTAC'20,'23]

Team Automata by Example

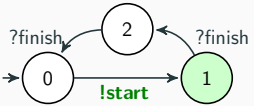
Team Automata: **not all system transitions are meaningful!**



$\mathcal{R}unner_1$



$\mathcal{R}unner_2$



Controller

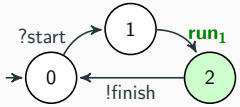
Team Automata

[CSCW'01'03] [FM'03,'21,'23] [TCS'12'13]

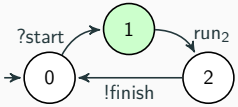
[COORDINATION'17,'20,'24] [ICTAC'20,'23]

Team Automata by Example

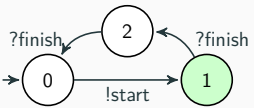
Team Automata: **not all system transitions are meaningful!**



$\mathcal{R}unner_1$



$\mathcal{R}unner_2$



Controller

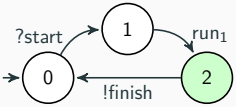
Team Automata

[CSCW'01'03] [FM'03,'21,'23] [TCS'12'13]

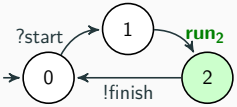
[COORDINATION'17,'20,'24] [ICTAC'20,'23]

Team Automata by Example

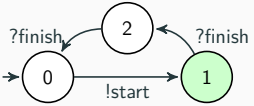
Team Automata: **not all system transitions are meaningful!**



$\mathcal{R}unner_1$



$\mathcal{R}unner_2$



Controller

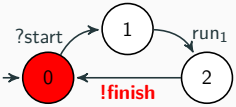
Team Automata

[CSCW'01'03] [FM'03,'21,'23] [TCS'12'13]

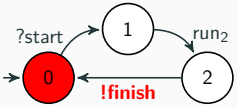
[COORDINATION'17,'20,'24] [ICTAC'20,'23]

Team Automata by Example

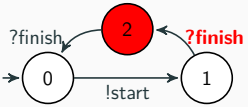
Team Automata: **not all system transitions are meaningful!**



$\mathcal{R}unner_1$



$\mathcal{R}unner_2$



Controller

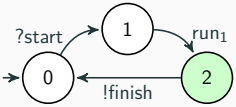
Team Automata

[CSCW'01'03] [FM'03,'21,'23] [TCS'12'13]

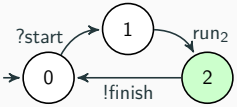
[COORDINATION'17,'20,'24] [ICTAC'20,'23]

Team Automata by Example

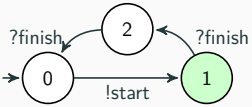
Team Automata: **not all system transitions are meaningful!**



$\mathcal{R}unner_1$



$\mathcal{R}unner_2$



Controller

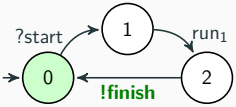
Team Automata

[CSCW'01'03] [FM'03,'21,'23] [TCS'12'13]

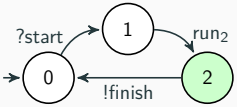
[COORDINATION'17,'20,'24] [ICTAC'20,'23]

Team Automata by Example

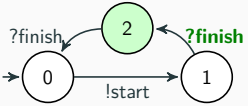
Team Automata: **not all system transitions are meaningful!**



$\mathcal{R}unner_1$



$\mathcal{R}unner_2$



Controller

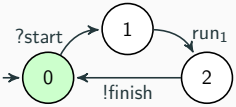
Team Automata

[CSCW'01'03] [FM'03,'21,'23] [TCS'12'13]

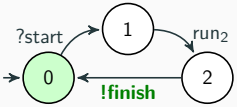
[COORDINATION'17,'20,'24] [ICTAC'20,'23]

Team Automata by Example

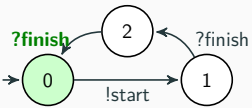
Team Automata: **not all system transitions are meaningful!**



$\mathcal{R}unner_1$



$\mathcal{R}unner_2$



Controller

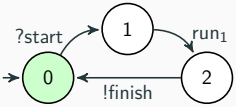
Team Automata

[CSCW'01'03] [FM'03,'21,'23] [TCS'12'13]

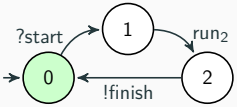
[COORDINATION'17,'20,'24] [ICTAC'20,'23]

Team Automata by Example

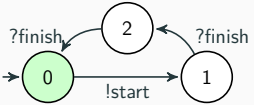
Extended Team Automata: Constrained Multi-party Synchronisations



$\mathcal{R}unner_1$



$\mathcal{R}unner_2$



Controller

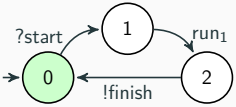
Extended TA synchronisations

[CSCW'01'03] [FM'03,'21,'23] [TCS'12'13]

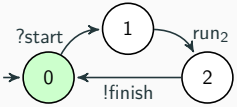
[COORDINATION'17,'20,'24] [ICTAC'20,'23]

Team Automata by Example

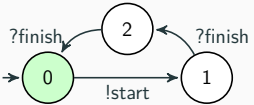
Extended Team Automata: Constrained Multi-party Synchronisations



$\mathcal{R}unner_1$



$\mathcal{R}unner_2$



Controller

Extended TA synchronisations

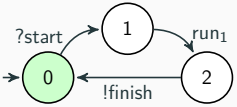
[CSCW'01'03] [FM'03,'21,'23] [TCS'12'13]

[COORDINATION'17,'20,'24] [ICTAC'20,'23]

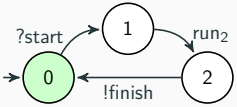
multi-party

$\text{Ctr} \rightarrow \{R1, R2\}$: start

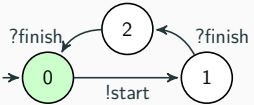
Extended Team Automata: Constrained Multi-party Synchronisations



$\mathcal{R}unner_1$



$\mathcal{R}unner_2$



Controller

Extended TA synchronisations

[CSCW'01'03] [FM'03,'21,'23] [TCS'12'13]

[COORDINATION'17,'20,'24] [ICTAC'20,'23]

multi-party

Ctr $\rightarrow \{R1, R2\}$: start

constrained

start: $1 \rightarrow 2$

finish: $1 \rightarrow 1$

APM 2025 $\rightarrow$ FM 2026:
Asynchronous Team Automata

Inspiration from multi-composition of asynchronous systems of CFSMs

D. Brand and P. Zafiropulo, On Communicating Finite-State Machines. *Journal of the ACM* 30 (1983)

F. Barbanera and R. Hennicker, Safe Composition of Systems of Communicating Finite State Machines @ ICE'24

Inspiration from multi-composition of asynchronous systems of CFSMs

D. Brand and P. Zafiropulo, On Communicating Finite-State Machines. *Journal of the ACM* 30 (1983)

F. Barbanera and R. Hennicker, Safe Composition of Systems of Communicating Finite State Machines @ ICE'24

Basically finite-state I/O-automata that communicate by asynchronous exchanges of messages via buffered FIFO channels, but

- CFSM use potentially infinite FIFO buffers as message channels (like MPST)
- CFSM systems use binary peer-to-peer communication with rich local actions

Behaviour of CFSM systems formalised as a transition relation on configurations (which are pairs of a tuple of component states and a tuple of channel buffers)

Inspiration from multi-composition of asynchronous systems of CFSMs

D. Brand and P. Zafiropulo, On Communicating Finite-State Machines. *Journal of the ACM* 30 (1983)

F. Barbanera and R. Hennicker, Safe Composition of Systems of Communicating Finite State Machines @ ICE'24

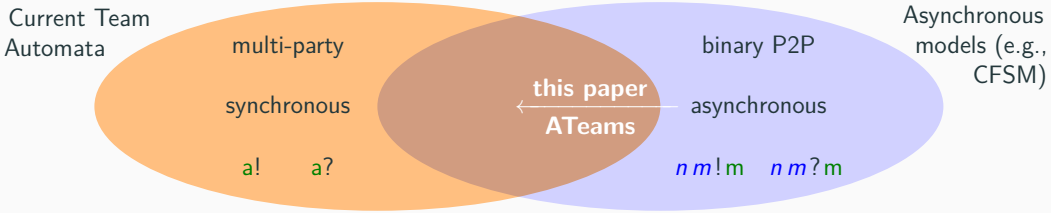
Basically finite-state I/O-automata that communicate by asynchronous exchanges of messages via buffered FIFO channels, but

- CFSM use potentially infinite FIFO buffers as message channels (like MPST)
- CFSM systems use binary peer-to-peer communication with rich local actions

Behaviour of CFSM systems formalised as a transition relation on configurations (which are pairs of a tuple of component states and a tuple of channel buffers)

Generalise to asynchronous multi-party communication for team automata (including receptiveness and responsiveness properties in asynchronous setting)

This paper's challenge:



"Every good talk should include a Venn diagram" – Einar (Lima, Peru, 2023)

Variations of the Race example in ATeams

```
// Synchronous Race
acts
 start:  1->2, sync;
 finish: 1->1, sync;
proc
 Ctr = start!.
       finish?.finish?.
       Ctr
 R = start?.finish!.R
init
 Ctr || R || R
```

[Link]

```
// Async. Race (sender)
acts
 start:  1->2, fifo@snd;
 finish: 2->1, fifo@snd;
proc
 Ctr = start!.
       finish?r1,r2.
       Ctr
 R = start?c.finish!.R
init
 c:Ctr || r1:R || r2:R
```

[Link]

```
// Async. Race (global)
acts
 start: 1->2, fifo@global;
 finish:2->1, fifo@global;
proc
 Ctr = start!.
       finish?.
       Ctr
 R = start?.finish!.R
init
 Ctr || R || R
```

[Link]

Syntax ATeams

$$P := K \mid \mathbf{0} \mid \alpha.P \mid P + P \qquad \text{(process)}$$

$$\alpha := \mathsf{a}! \mid \mathsf{a}? \mid \mathsf{a}!\overline{n} \mid \mathsf{a}?\overline{n} \qquad \text{(action)}$$

$$st := (\mathsf{sync}, intrv, intrv) \mid (\mathsf{async}, intrv, intrv, buft, loct) \qquad \text{(synchronisation type)}$$

$$intrv := (i, i) \mid (i, \infty) \qquad \text{(interval)}$$

$$buft := \mathsf{fifo} \mid \mathsf{bag} \mid \ldots \qquad \text{(buffer type)}$$

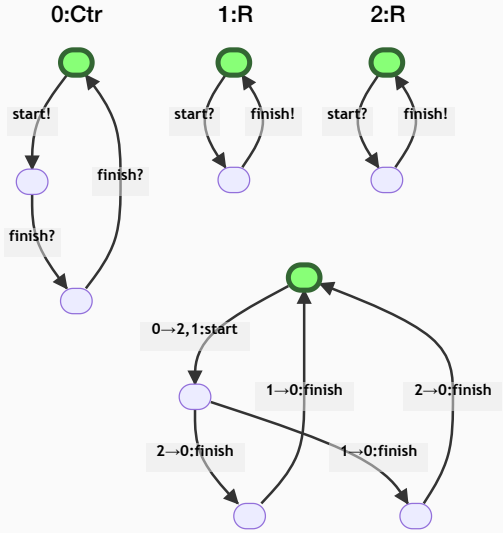
$$loct := \mathsf{@snd} \mid \mathsf{@rcv} \mid \mathsf{@global} \mid \mathsf{@snd\text{-}rcv} \qquad \text{(location type)}$$

A **System** consists of

Process' definitions + Action's types + Agents (processes)

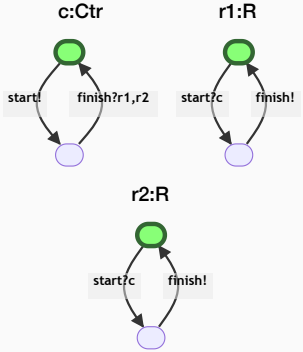
Semantics ATeams: synchronous communication

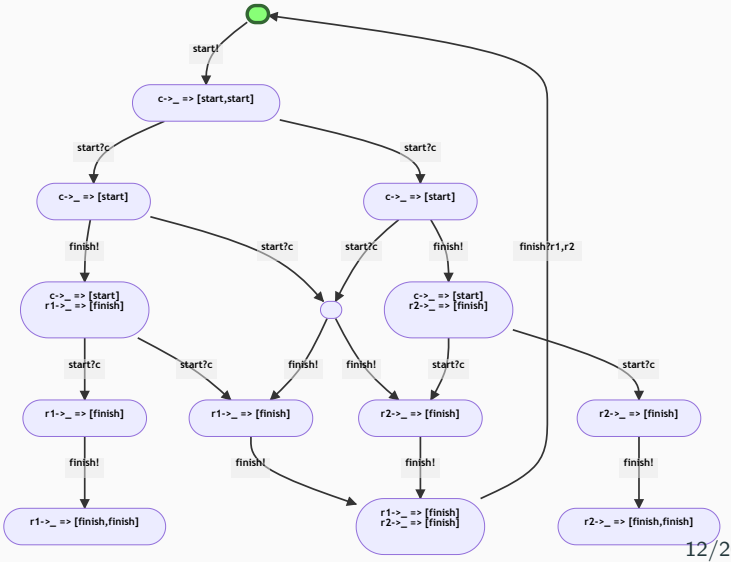
```
// Synchronous Race
acts
 start:  1->2, sync;
 finish: 1->1, sync;
proc
 Ctr = start!.
       finish?.finish?.
       Ctr
 R = start?.finish!.R
init
 Ctr ‖ R ‖ R
```



Semantics ATeams: asynchronous communication (sending)

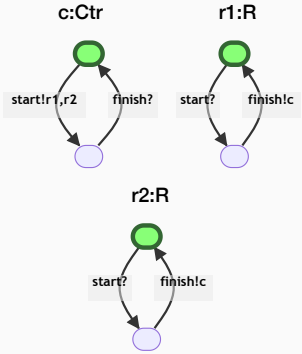
```
// Async. Race (sender)
acts
 start:  1->2, fifo@snd;
 finish: 2->1, fifo@snd;
```

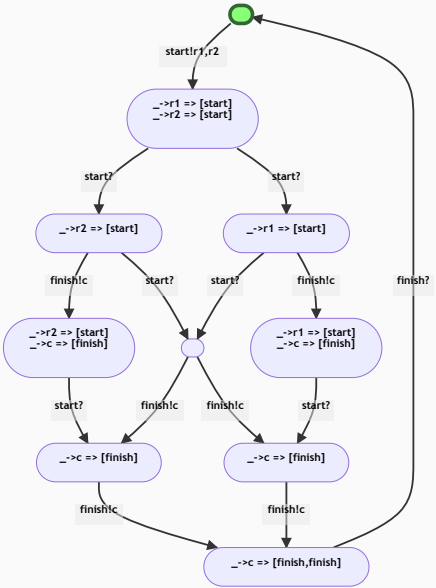




Semantics ATeams: asynchronous communication (receiving)

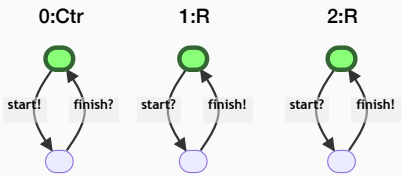
```
// Async. Race (receiver)
acts
 start:  1->2, fifo@rcv;
 finish: 2->1, fifo@rcv;
```



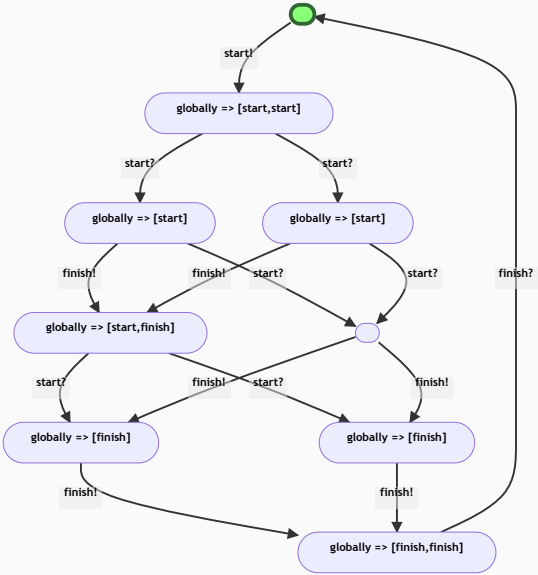


Semantics ATeams: asynchronous communication (global)

```
// Async. Race (global)
acts
 start:  1->2, fifo@global;
 finish: 2->1, fifo@global;
```



1:R can start twice before 2:R



Principle 1: *Asynchronous sending messages should not block.*

Principle 2: *The sender or receiver of an asynchronous action must not be explicitly mentioned when this is disregarded by the buffer location.*

Principle 3: *The number of asynchronous messages sent or received must be deterministic.*

Well-formed and well-behaved ATeams

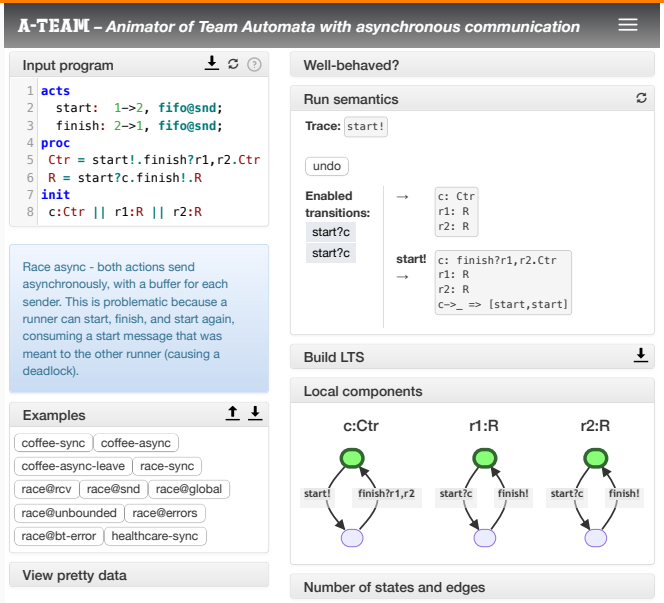
Well-formedness

Syntactic checks: violations to the synchronisation types
(e.g., wrong numbers of targets or sources of an action)

Well-behavedness

Semantic checks: from traversing the full state-space
(e.g., deadlock or orphan messages found)

A-Team: Animator of ATeams



Feedback in A-Team (ill-formed)

Input program

```
1 acts start:  1->2, sync;
2      finish: 2->1, fifo@snd;
3 proc Ctr = start!.finish?r1.Ctr
4      R = start?.finish!c.R
5 init c:Ctr || r1:R || r2:R
```

Error raised by 'Well-behaved?'': Trying to send 'finish!c' to {c} but there is no guarantees that the receivers are the correct ones. You should either remove the explicit set of senders, or change the synchronisation type.

[Ctr] Trying to receive 'finish' from {r1} but expected #{r1} ∈ {2}.

[R] Sending finish to c, but no destination should be specified.

Well-behaved?

Try A-Team yourself: https://fm-dcc.github.io/a-team/

Feedback in A-Team (ill-behaved)

Input program

```
1  acts
2    default: fifo@rcv, 1->1;
3    coin; coffee;
4
5  proc
6   // User may leave
7   Usr  = coin!m.coffee?.(leave+Usr)
8   Mach = coin?.coffee!u.Mach
9   // Add an extra coin
10  Usr2 = coin!m.Usr
11
12 init
13  u:Usr2||m:Mach
```

Variation of the asynchronous coffee machine with an extra coin and a terminating option, to create orphan messages.

Examples

Hide comm-props | Max buffers' sizes

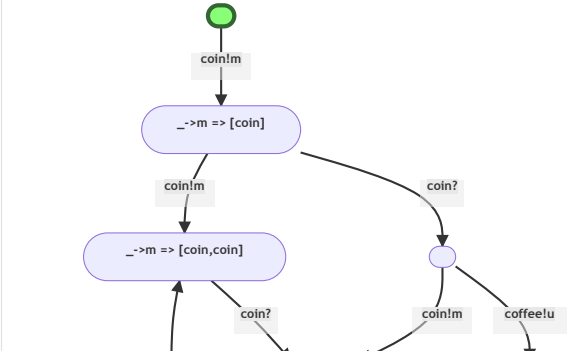
View pretty data

Well-behaved?

```
Deadlock found: {u: skip, m: Mach, _->u=>[coffee]}
[orphan-messages] Agent u terminated and has incomming messages: {u: skip, m: Mach, _->u=>[coffee]}
[strong-responsiveness] m can receive coin, but the system cannot (yet) send it: {u: skip, m: Mach,
[strong-responsiveness] m can receive coin, but the system cannot (yet) send it: {u: coffee?.(leave+
[strong-responsiveness] u can receive coffee, but the system cannot (yet) send it: {u: coffee?.(leav
```

Run semantics

Build LTS



Conclusion and Publicity

Thanks for your attention!

It would be great to meet some of you at FM 2026:

- Conference: May 18th–22nd
- New at FM: track on Tests and Proofs (TAP)
- Co-located: 18th International Conference on Rigorous State Based Methods (ABZ)

https://conf.researchr.org/home/fm-2026