Can we Communicate?
Using Dynamic Logic to Verify Featured Team Automata

Maurice H. ter Beek[1] Guillermina Cledou[2] Rolf Hennicker[3] José Proença[4]

[1]ISTI–CNR, Pisa, Italy

[2]University of Minho, Portugal

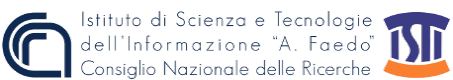
[3]LMU Munich, Germany

[4]Polytechnic Institute of Porto, Portugal

Pre-VaMoS Workshop at ITU, Copenhagen, 24 January 2023

Before we start, and I forget . . .

The Formal Methods and Tools lab at ISTI–CNR in Pisa, Italy, offers 1 grant for a 3-year PhD programme at the University of Pisa on *"Formal modelling and analysis of critical software systems"*, financed by the PNRR project *"Railway Transport"* of the Sustainable Mobility Center



The deadline for applications is: **February 1st, 2023, at 1pm**

Requirement for participation: master's degree by February 28th

If you're interested, feel free to contact me for more information!

References

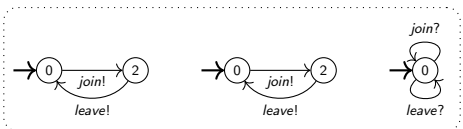
FM21 M.H. ter Beek, G. Cledou, R. Hennicker & J. Proença, *Featured Team Automata*. In Proceedings of the 24th International Symposium on Formal Methods (FM'21) (M. Huisman, C. Păsăreanu, and N. Zhan, eds.), Lecture Notes in Computer Science 13047, Springer, 2023.

FM23 M.H. ter Beek, G. Cledou, R. Hennicker & J. Proença, *Can we Communicate? Using Dynamic Logic to Verify Team Automata*. In Proceedings of the 25th International Symposium on Formal Methods (FM'23) (M. Chechik, J.-P. Katoen, and M. Leucker, eds.), Lecture Notes in Computer Science 14000, Springer, 2023.

Background

Team Automata:

- Systems of communicating components: synchronise over shared actions
- Synchronisation types per action: *peer-2-peer*, *broadcast*, ...



Goal: safe communication – *no message loss, no indefinite waiting, ...*

---

ter Beek, *Team Automata*. Ph.D. thesis, Leiden University, 2003

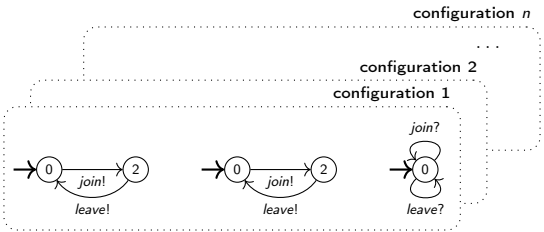
ter Beek, Ellis, Kleijn & Rozenberg, *Synchronizations in team automata for groupware systems*. Comput. Sup. Coop. Work 12, 2003

ter Beek, Hennicker & Kleijn, *Compositionality of Safe Communication in Systems of Team Automata*. ICTAC 2020

Motivation

Many systems today are highly configurable (in terms of features):

- Large sets of similar systems that share a lot of behaviour but differ in other



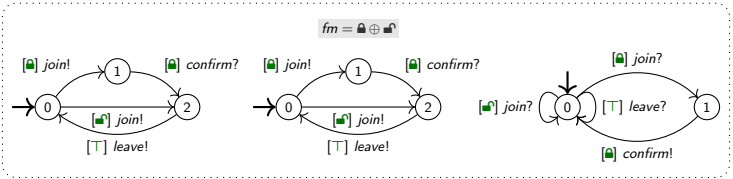
Challenge: system-by-system analysis of safe communication quickly becomes unfeasible

Classen, Cordy, Schobbens, Heymans, Legay & Raskin, *Featured Transition Systems: Foundations for Verifying Variability-Intensive Systems and Their Application to LTL Model Checking*. IEEE Trans. Softw. Eng. 39, 2013

Approach

Featured Team Automata:

- Families (sets) of Team Automata model as a Software Product Line
- Single model parametrised by features (e.g.: 🔒, 🔓), and a feature model (🔒 ⊕ 🔓)

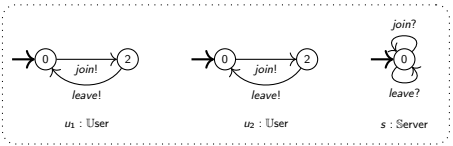


Goal: family-based analysis of safe communication

ter Beek, Cledou, Hennicker & Proença, *Featured Team Automata*. FM 2021

Team Automata

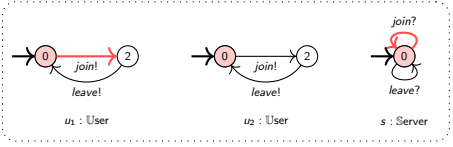
ICTAC'20:



Systems

Team Automata

ICTAC'20:

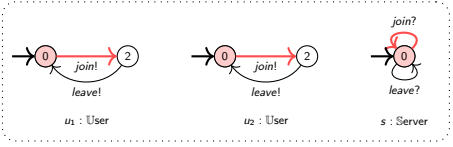


Systems

$$(0,0,0) \xrightarrow{(\{u_1\},join,\{s\})} (2,0,0)$$

Team Automata

ICTAC'20:

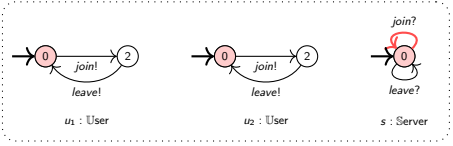


Systems

$$(0,0,0) \xrightarrow{(\{u_1,u_2\},join,\{s\})} (2,2,0)$$

Team Automata

ICTAC'20:



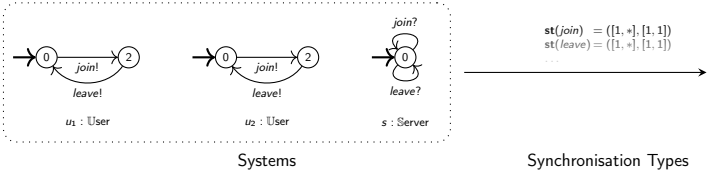
Systems

$$(0,0,0) \xrightarrow{(\{\},join,\{s\})} (0,0,0)$$

might not be desirable

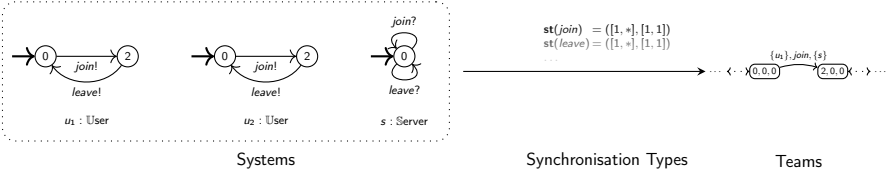
Team Automata

ICTAC'20:



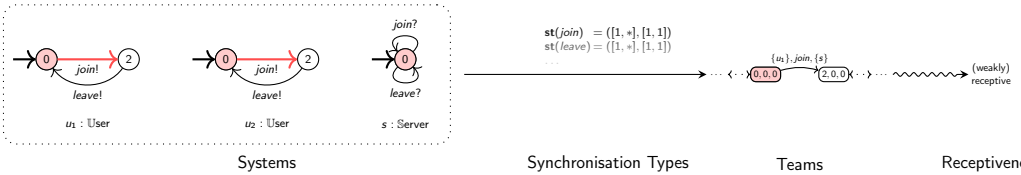
Team Automata

ICTAC'20:



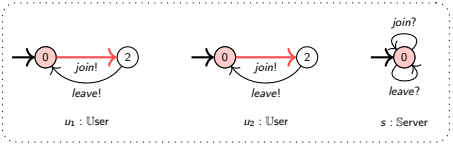
Team Automata

ICTAC'20:

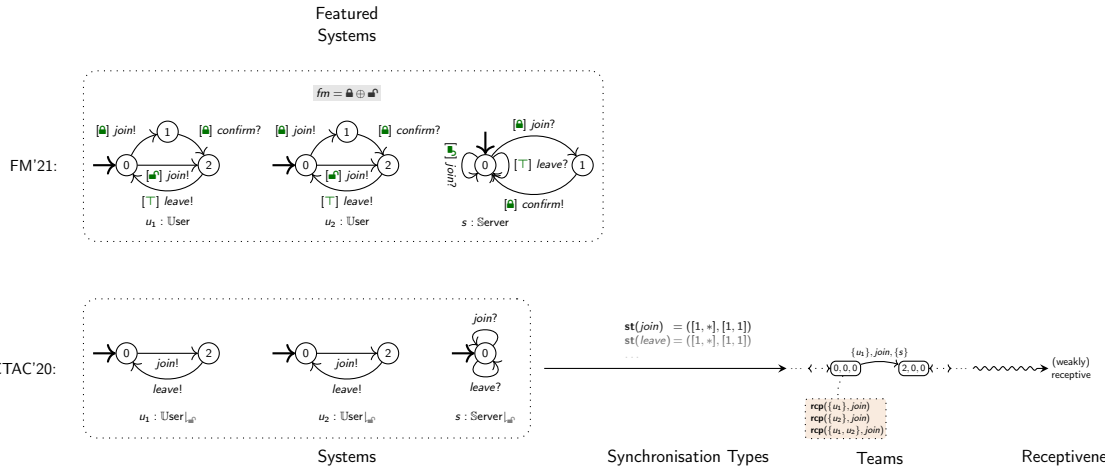


Team Automata

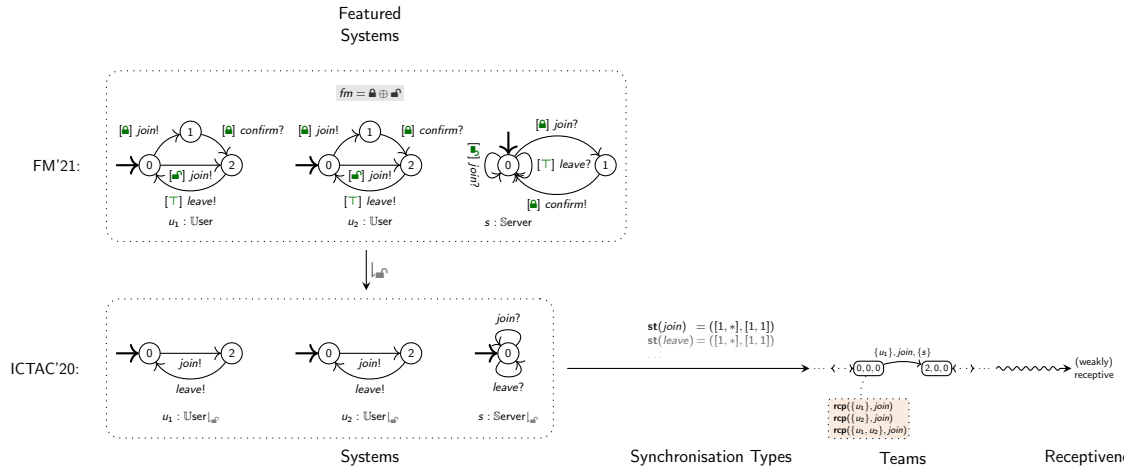
ICTAC'20:



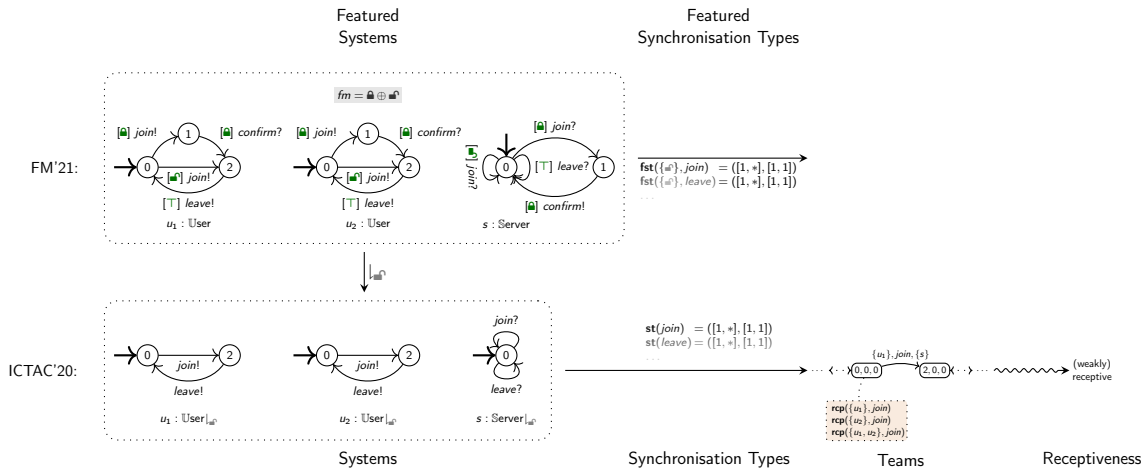
Featured Team Automata



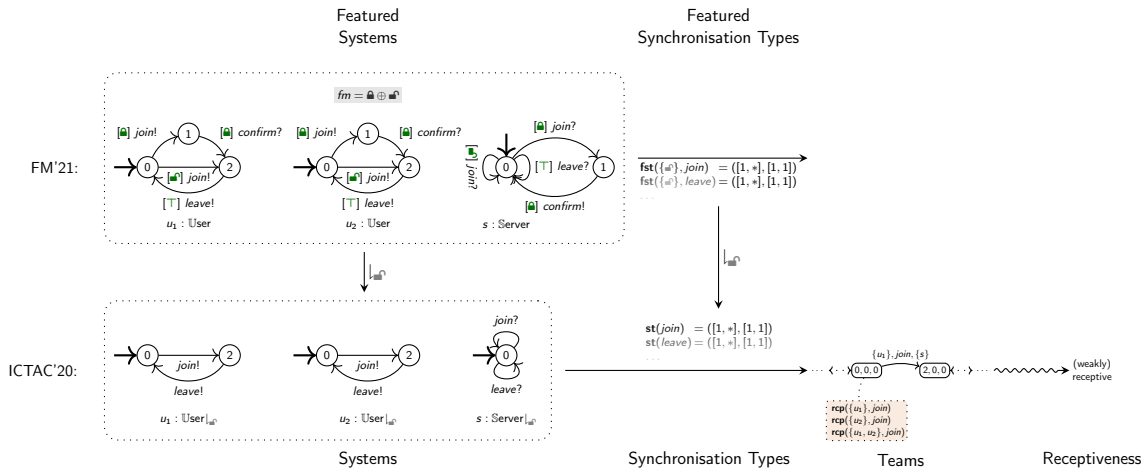
Featured Team Automata



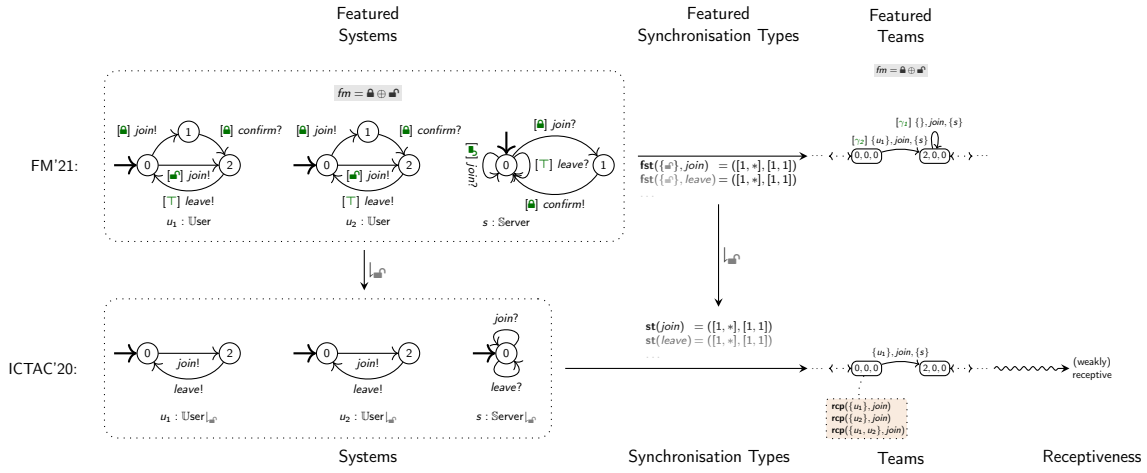
Featured Team Automata



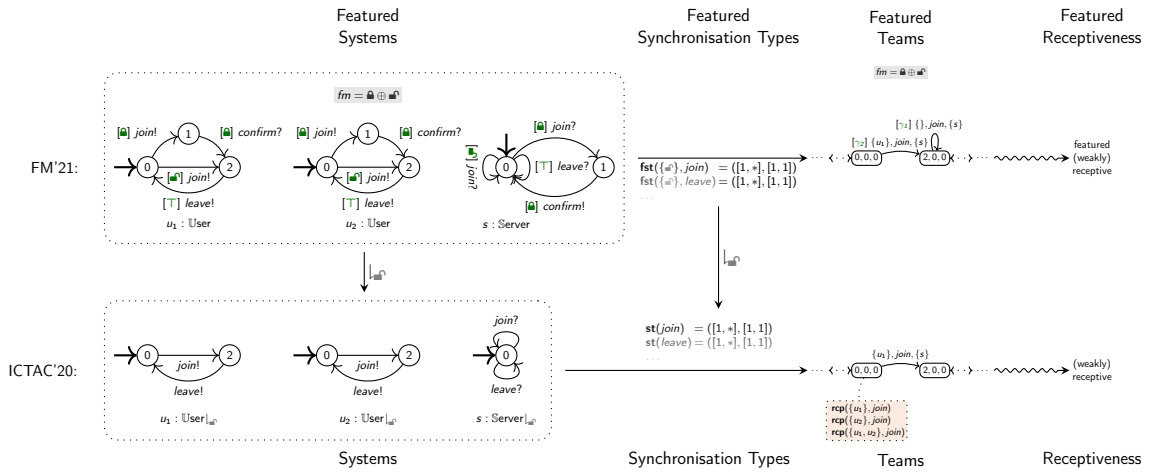
Featured Team Automata



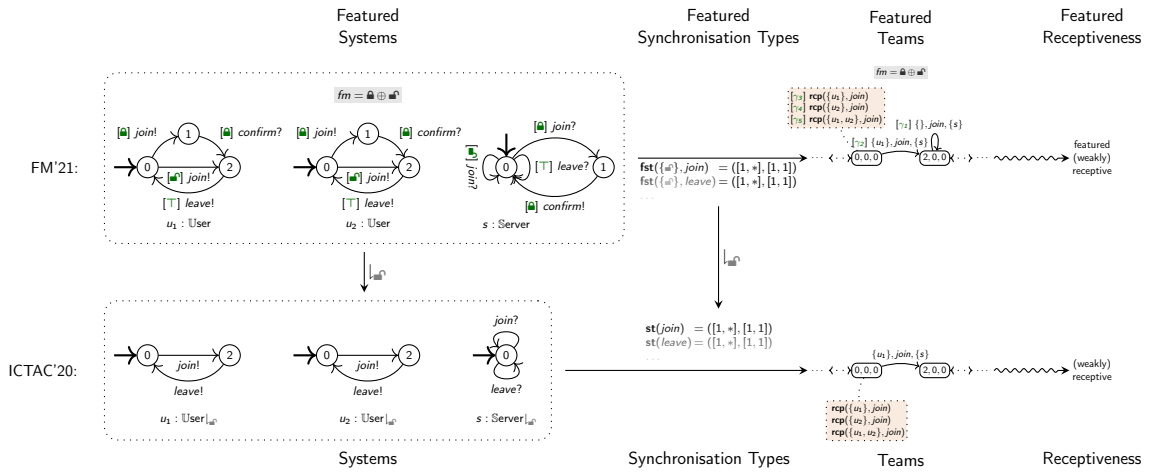
Featured Team Automata



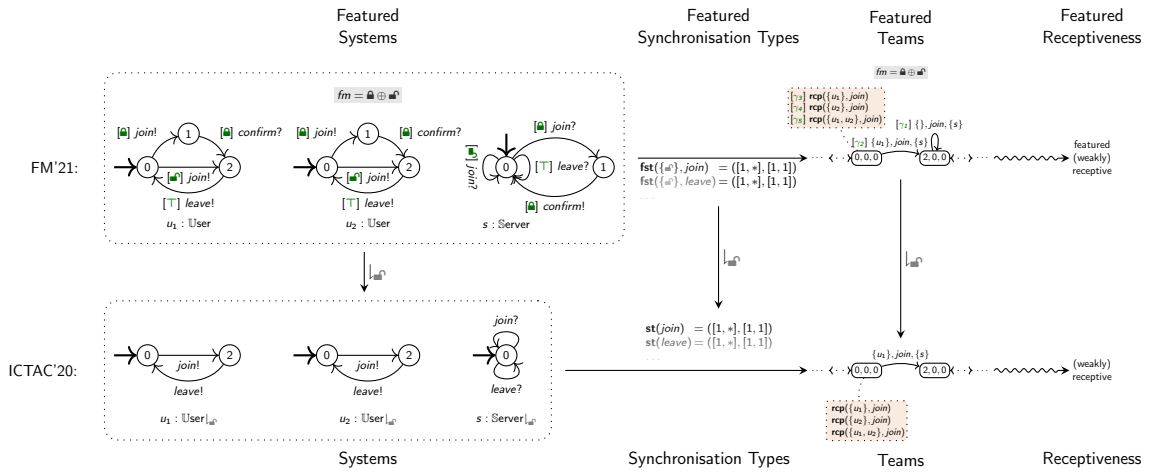
Featured Team Automata



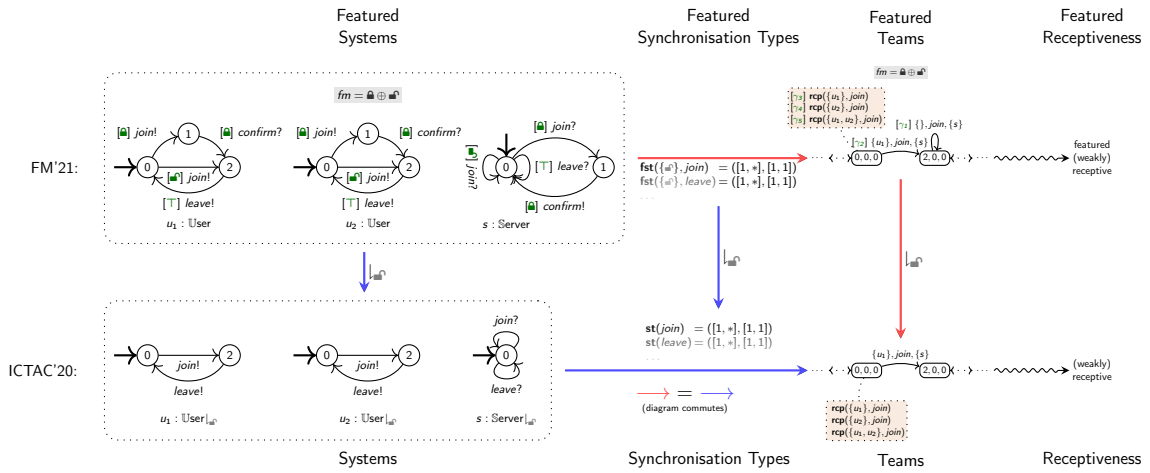
Featured Team Automata



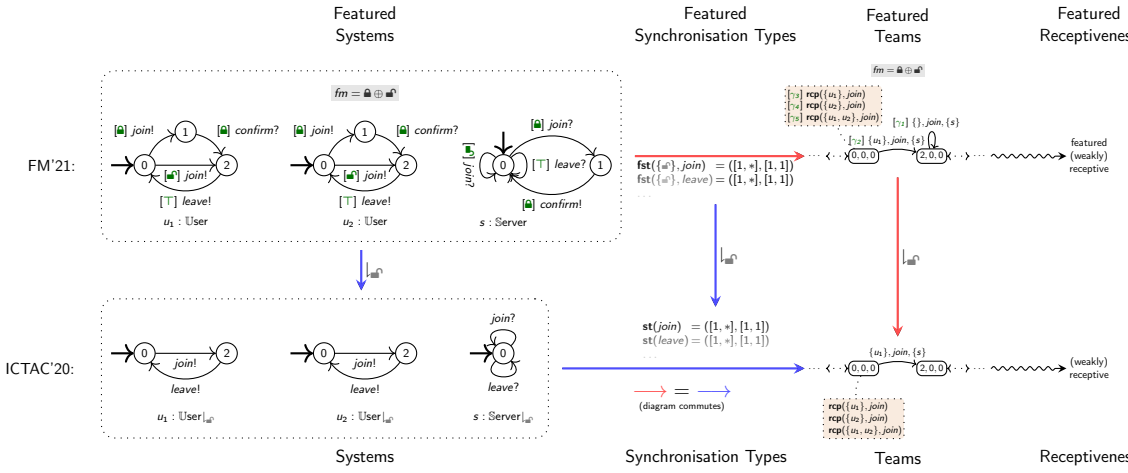
Featured Team Automata



Featured Team Automata

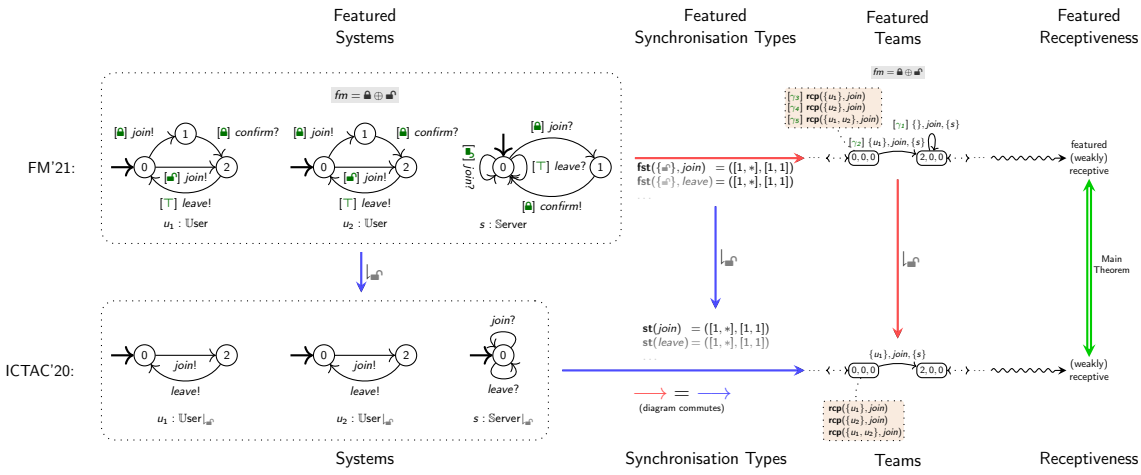


Featured Team Automata



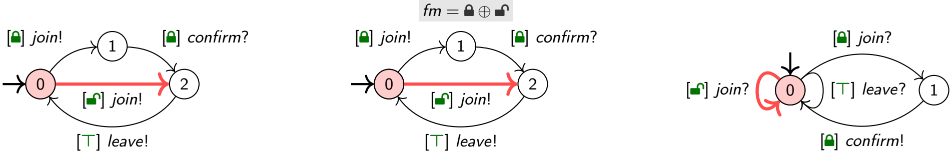
Online prototype: `http://arcatools.org/feta`

Featured Team Automata



Online prototype: `http://arcatools.org/feta`

Featured Team Automata Transitions



$u_1 : \mathbb{U}\text{ser}$ $\quad$ $u_2 : \mathbb{U}\text{ser}$ $\quad$ $s : \mathbb{S}\text{erver}$

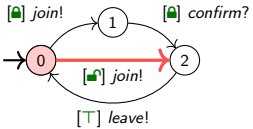
Transitions are constrained with feature expressions by:

- local feature expressions: characterise the products with all local transitions present
- **fst**: characterise the products that satisfy the corresponding synchronisation type

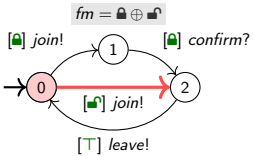
$$\mathbf{fst}(\{🔒\}, join) = ([1,1],[1,1]) \qquad \mathbf{fst}(\{🔓\}, join) = ([1,*],[1,1])$$

$$(0,0,0) \xrightarrow{[\qquad](\{u_1,u_2\}, join, \{s\})}_{\mathbf{fst}[\mathcal{S}]} (2,2,0)$$

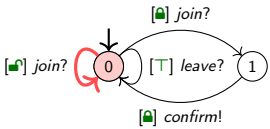
Featured Team Automata Transitions



$u_1 : \mathbb{U}\text{ser}$



$u_2 : \mathbb{U}\text{ser}$



$s : \mathbb{S}\text{erver}$

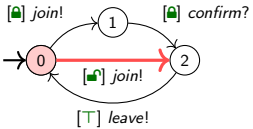
Transitions are constrained with feature expressions by:

- local feature expressions: characterise the products with all local transitions present
- **fst**: characterise the products that satisfy the corresponding synchronisation type

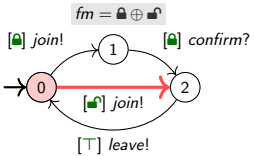
$$\textbf{fst}(\{🔒\}, join) = ([1,1],[1,1]) \qquad \textbf{fst}(\{🔓\}, join) = ([1,*],[1,1])$$

$$(0,0,0) \xrightarrow{[🔓 \wedge 🔒 \wedge 🔓 \quad](\{u_1,u_2\}, join, \{s\})}_{\textbf{fst}[\mathcal{S}]} (2,2,0)$$

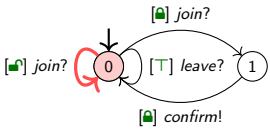
Featured Team Automata Transitions



$u_1 : \mathbb{U}\text{ser}$



$u_2 : \mathbb{U}\text{ser}$



$s : \mathbb{S}\text{erver}$

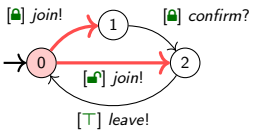
Transitions are constrained with feature expressions by:

- local feature expressions: characterise the products with all local transitions present
- **fst**: characterise the products that satisfy the corresponding synchronisation type

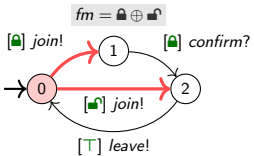
$$\textbf{fst}(\{🔒\}, join) = ([1,1],[1,1]) \qquad \textbf{fst}(\{🔓\}, join) = ([1,*],[1,1])$$

$$(0,0,0) \xrightarrow{[🔓\wedge🔓\wedge🔓\wedge🔓\wedge\neg🔒](\{u_1,u_2\},join,\{s\})}_{\textbf{fst}[\mathcal{S}]} (2,2,0)$$

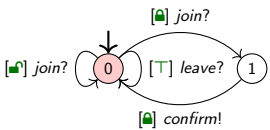
Featured Receptiveness Requirements



$u_1 : \mathbb{U}\text{ser}$



$u_2 : \mathbb{U}\text{ser}$



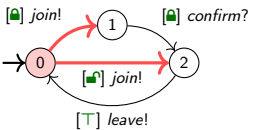
$s : \mathbb{S}\text{erver}$

$$\mathbf{fst}(\{🔒\}, join) = ([1,1],[1,1]) \qquad \mathbf{fst}(\{🔓\}, join) = ([1,*],[1,1])$$

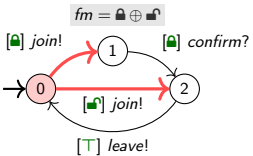
At state $(0,0,0)$:

[ ] $\mathbf{rcp}(\{u_1\}, join)$ $\wedge$ [ ] $\mathbf{rcp}(\{u_2\}, join)$ $\wedge$ [ ] $\mathbf{rcp}(\{u_1, u_2\}, join)$

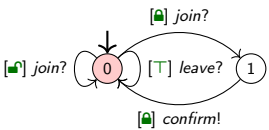
Featured Receptiveness Requirements



$u_1 : \mathbb{U}\text{ser}$



$u_2 : \mathbb{U}\text{ser}$



$s : \mathbb{S}\text{erver}$

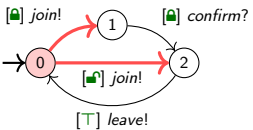
$$\mathbf{fst}(\{🔒\}, join) = ([1,1],[1,1]) \qquad \mathbf{fst}(\{🔓\}, join) = ([1,*],[1,1])$$

At state $(0,0,0)$:

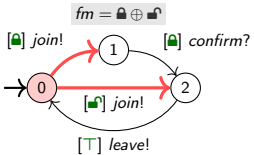
[ ] $\mathbf{rcp}(\{u_1\}, join)$ $\wedge$ [ ] $\mathbf{rcp}(\{u_2\}, join)$ $\wedge$ [ ] $\mathbf{rcp}(\{u_1, u_2\}, join)$

Featured receptiveness requirements are constrained with feature expression by:

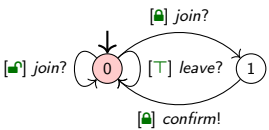
Featured Receptiveness Requirements



$u_1 : \mathbb{U}\text{ser}$



$u_2 : \mathbb{U}\text{ser}$



$s : \mathbb{S}\text{erver}$

$$\mathbf{fst}(\{🔒\}, join) = ([1,1],[1,1]) \qquad \mathbf{fst}(\{🔓\}, join) = ([1,*],[1,1])$$

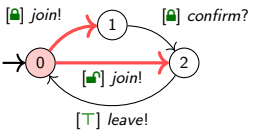
At state $(0,0,0)$:

[ ] $\mathbf{rcp}(\{u_1\}, join)$ $\wedge$ [ ] $\mathbf{rcp}(\{u_2\}, join)$ $\wedge$ [ ] $\mathbf{rcp}(\{u_1, u_2\}, join)$

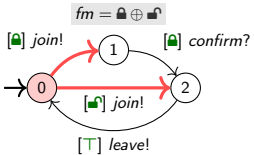
Featured receptiveness requirements are constrained with feature expression by:

- local feature expressions: characterise products with enabled local transitions

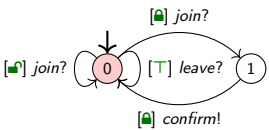
Featured Receptiveness Requirements



$u_1 : \mathbb{U}\text{ser}$



$u_2 : \mathbb{U}\text{ser}$



$s : \mathbb{S}\text{erver}$

$$\mathbf{fst}(\{🔒\}, join) = ([1,1],[1,1]) \qquad \mathbf{fst}(\{🔓\}, join) = ([1,*],[1,1])$$

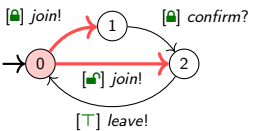
At state $(0,0,0)$:

[🔒 ∨ 🔓] $\mathbf{rcp}(\{u_1\}, join)$ ∧ [ ] $\mathbf{rcp}(\{u_2\}, join)$ ∧ [ ] $\mathbf{rcp}(\{u_1, u_2\}, join)$

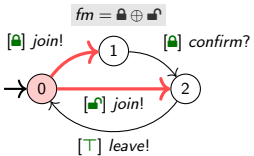
Featured receptiveness requirements are constrained with feature expression by:

- local feature expressions: characterise products with enabled local transitions

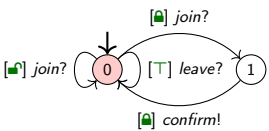
Featured Receptiveness Requirements



$u_1 : \mathbb{U}\text{ser}$



$u_2 : \mathbb{U}\text{ser}$



$s : \mathbb{S}\text{erver}$

$$\mathbf{fst}(\{🔒\}, join) = ([1,1],[1,1]) \qquad \mathbf{fst}(\{🔓\}, join) = ([1,*],[1,1])$$

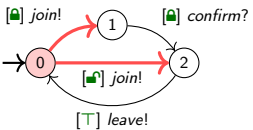
At state $(0,0,0)$:

[🔒 ∨ 🔓] $\mathbf{rcp}(\{u_1\}, join)$ ∧ [ ] $\mathbf{rcp}(\{u_2\}, join)$ ∧ [ ] $\mathbf{rcp}(\{u_1, u_2\}, join)$

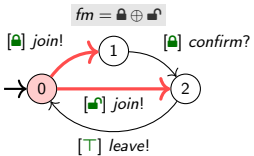
Featured receptiveness requirements are constrained with feature expression by:

- local feature expressions: characterise products with enabled local transitions
- **fst**: characterise products with the correct number of senders

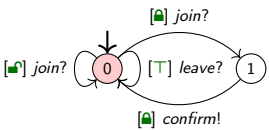
Featured Receptiveness Requirements



$u_1 : \mathbb{U}\text{ser}$



$u_2 : \mathbb{U}\text{ser}$



$s : \mathbb{S}\text{erver}$

$$\textbf{fst}(\{🔒\}, join) = ([1,1],[1,1]) \qquad \textbf{fst}(\{🔓\}, join) = ([1,*],[1,1])$$

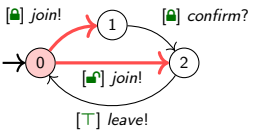
At state $(0,0,0)$:

$[🔒 \vee 🔓 \wedge fm]\ \textbf{rcp}(\{u_1\}, join)\ \wedge\ [\quad]\ \textbf{rcp}(\{u_2\}, join)\ \wedge\ [\qquad\qquad]\ \textbf{rcp}(\{u_1, u_2\}, join)$

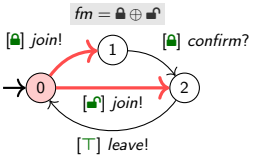
Featured receptiveness requirements are constrained with feature expression by:

- local feature expressions: characterise products with enabled local transitions
- **fst**: characterise products with the correct number of senders

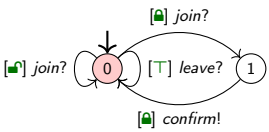
Featured Receptiveness Requirements



$u_1 : \mathbb{U}\text{ser}$



$u_2 : \mathbb{U}\text{ser}$



$s : \mathbb{S}\text{erver}$

$$\mathbf{fst}(\{🔒\}, join) = ([1,1],[1,1]) \qquad \mathbf{fst}(\{🔓\}, join) = ([1,*],[1,1])$$

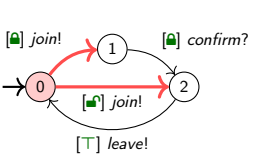
At state $(0,0,0)$:

$[🔒 \vee 🔓 \wedge fm]\ \mathbf{rcp}(\{u_1\}, join)\ \wedge\ [\quad]\ \mathbf{rcp}(\{u_2\}, join)\ \wedge\ [\qquad\qquad]\ \mathbf{rcp}(\{u_1, u_2\}, join)$

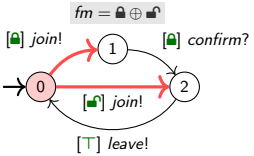
Featured receptiveness requirements are constrained with feature expression by:

- local feature expressions: characterise products with enabled local transitions
- **fst**: characterise products with the correct number of senders
- reachable states: characterise products where the state is reachable

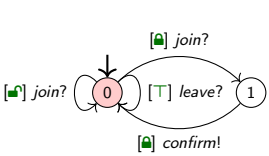
Featured Receptiveness Requirements



$u_1 : \mathbb{U}\text{ser}$



$u_2 : \mathbb{U}\text{ser}$



$s : \mathbb{S}\text{erver}$

$$\textbf{fst}(\{🔒\}, join) = ([1,1],[1,1]) \qquad \textbf{fst}(\{🔓\}, join) = ([1,*],[1,1])$$

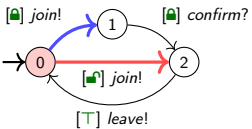
At state $(0,0,0)$:

$$[🔒 \vee 🔓 \wedge fm]\ \textbf{rcp}(\{u_1\}, join)\ \wedge\ [fm]\ \textbf{rcp}(\{u_2\}, join)\ \wedge\ [🔒 \vee 🔓 \wedge 🔓 \wedge \neg 🔒 \wedge fm]\ \textbf{rcp}(\{u_1, u_2\}, join)$$

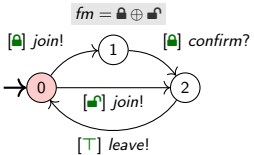
Featured receptiveness requirements are constrained with feature expression by:

- local feature expressions: characterise products with enabled local transitions
- **fst**: characterise products with the correct number of senders
- reachable states: characterise products where the state is reachable

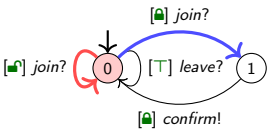
Compliance with requirements



$u_1 : \mathbb{U}\text{ser}$



$u_2 : \mathbb{U}\text{ser}$



$s : \mathbb{S}\text{erver}$

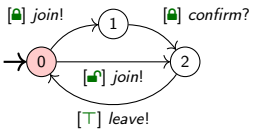
At state $(0,0,0)$:

$$\underline{[fm]\ \mathbf{rcp}(\{u_1\}, join)} \ \wedge\ [fm]\ \mathbf{rcp}(\{u_2\}, join) \ \wedge\ [🔓 \wedge \neg 🔒]\ \mathbf{rcp}(\{u_1, u_2\}, join)$$

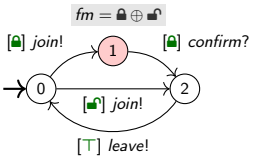
$$\{🔒\} : \ (0,0,0) \xrightarrow{[🔒 \wedge fm]\ (\{u_1\}, join, \{s\})} \mathbf{fst}[\mathcal{S}](1,0,1)$$

$$\{🔓\} : \ (0,0,0) \xrightarrow{[🔓 \wedge fm]\ (\{u_1\}, join, \{s\})} \mathbf{fst}[\mathcal{S}](2,0,0)$$

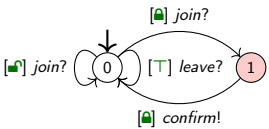
Compliance with requirements



$u_1 : \mathbb{U}\text{ser}$



$u_2 : \mathbb{U}\text{ser}$

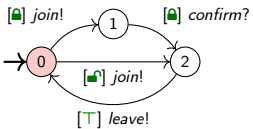


$s : \mathbb{S}\text{erver}$

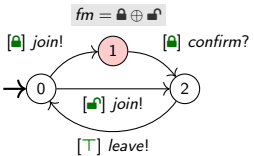
At state $(0, 1, 1)$:

$$[🔒 \wedge \neg 🔓]\ \mathbf{rcp}(\{u_1\}, join)\ \wedge\ \ldots$$

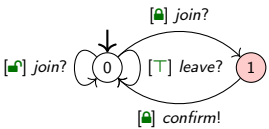
Compliance with requirements



$u_1 : \mathbb{U}\text{ser}$



$u_2 : \mathbb{U}\text{ser}$

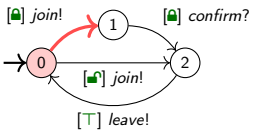


$s : \mathbb{S}\text{erver}$

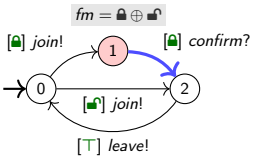
At state $(0, 1, 1)$:

$$\text{✗}[🔒 \wedge \neg 🔓]\ \mathbf{rcp}(\{u_1\}, join)\ \wedge\ \ldots$$

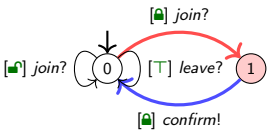
Weak compliance with requirements



$u_1 : \mathbb{U}\text{ser}$



$u_2 : \mathbb{U}\text{ser}$



$s : \mathbb{S}\text{erver}$

At state $(0, 1, 1)$:

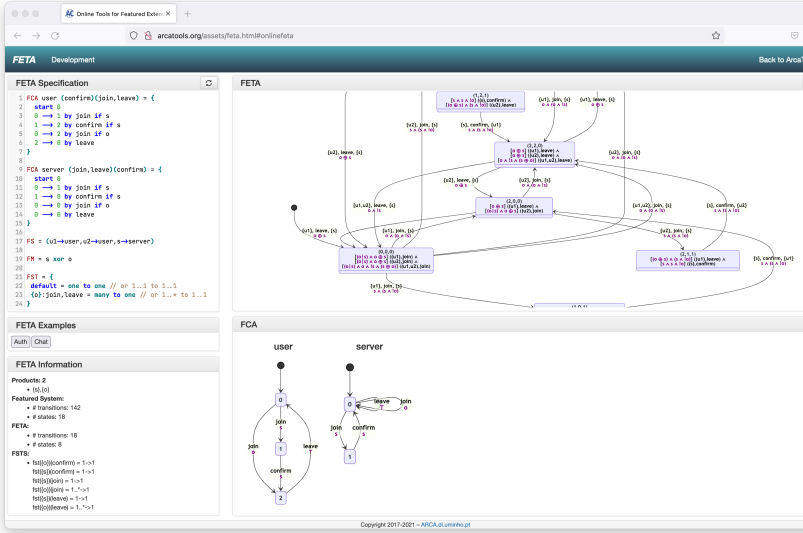
$$[🔒 \wedge \neg 🔓]\ \mathbf{rcp}(\{u_1\}, join)\ \wedge\ \ldots$$

$$\{🔒\} :\ (0,1,1) \xrightarrow{[🔒 \wedge \underline{fm}](\{s\}, confirm, \{u_2\})} \mathbf{fst}[\mathcal{S}](0,2,0) \xrightarrow{[🔒 \wedge \underline{fm}](\{u_1\}, join, \{s\})} \mathbf{fst}[\mathcal{S}](1,2,1)$$

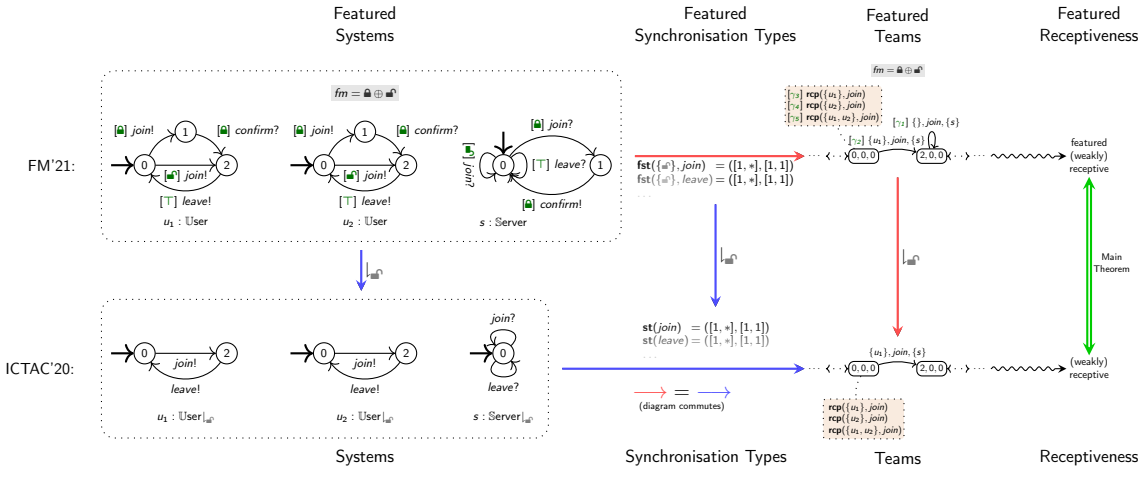
Online prototype

- Specify
- Generate*
- Visualise
- Statistics

*SAT solver to solve *fm*

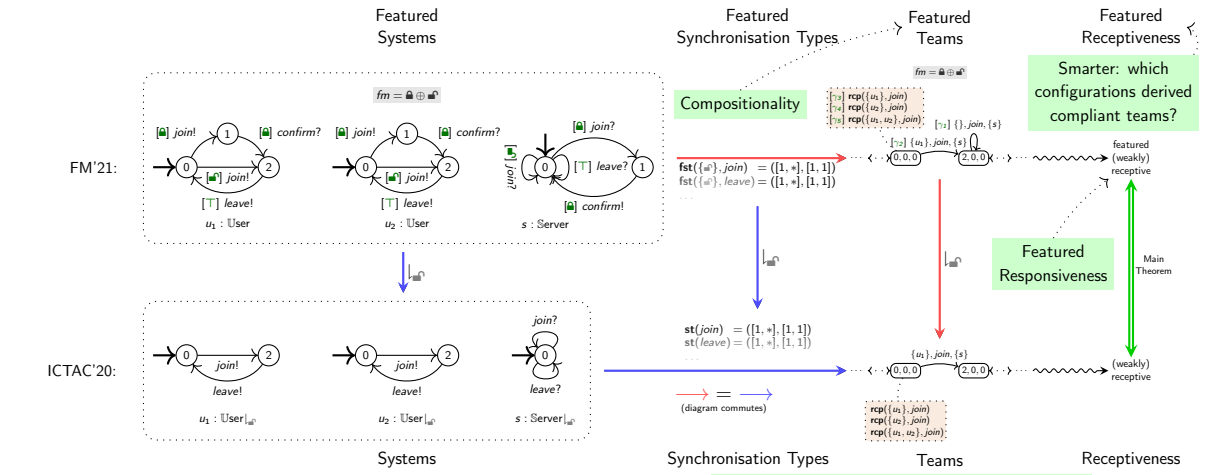


Wrapping up



Online prototype: `http://arcatools.org/feta`

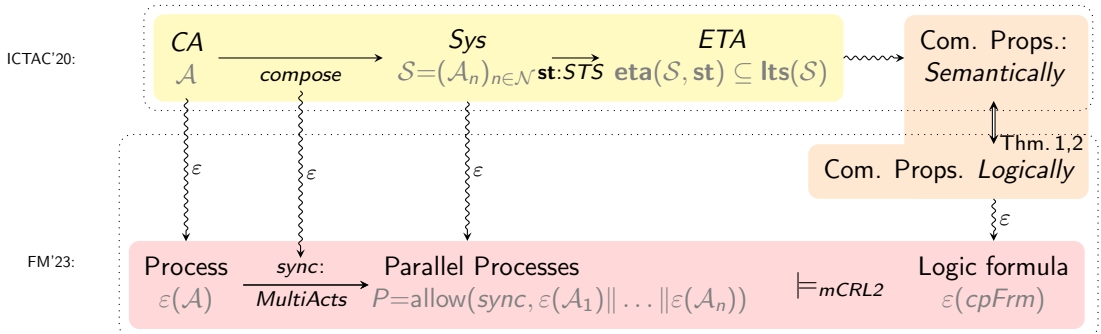
Future and ongoing work



Online prototype: `http://arcatools.org/feta`

Extentions (e.g. FM'23: generate requirements as PDL formulae and check compliance with mCRL2)

Using Dynamic Logic to Verify Team Automata



Online prototype: https://github.com/arcalab/team-a

ETA Specification

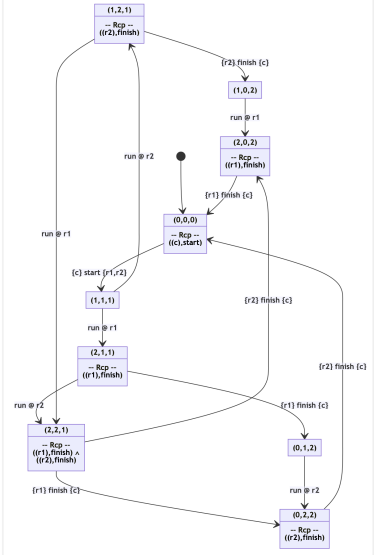
```
1  //Race example
2  CA runner (start)
3                (finish) = {
4    start 0
5    0 --> 1 by start
6    1 --> 2 by run
7    2 --> 0 by finish
8  }
9  CA controller (finish)
10                 (start) = {
11   start 0
12   0 --> 1 by start
13   1 --> 2 by finish
14   2 --> 0 by finish
15 }
16 S = (r1:runner, r2:runner,
17      c:controller)
18 STS = {
19   default = 1 to 1
20   start = 1 to 2
21 }
```

Race example

ETA Examples

Simple Race Chat

ETA diagram



Online prototype: https://github.com/arcalab/team-a

ETA Specification

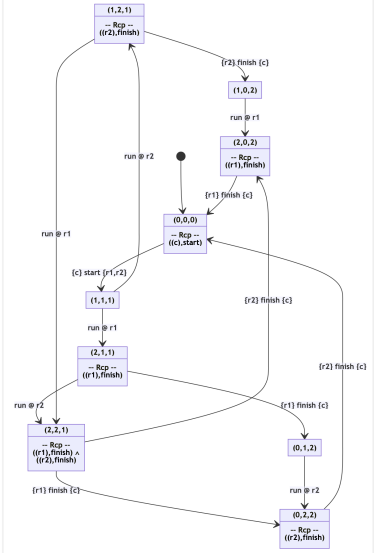
```
1  //Race example
2  CA runner (start)
3             (finish) = {
4   start 0
5   0 --> 1 by start
6   1 --> 2 by run
7   2 --> 0 by finish
8  }
9  CA controller (finish)
10            (start) = {
11  start 0
12  0 --> 1 by start
13  1 --> 2 by finish
14  2 --> 0 by finish
15 }
16 S = (r1:runner, r2:runner,
17      c:controller)
18 STS = {
19 default = 1 to 1
20 start = 1 to 2
21 }
```

Race example

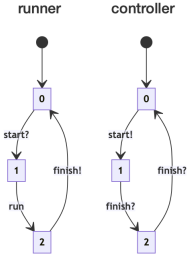
ETA Examples

Simple | Race | Chat

ETA diagram



CA



Online prototype: https://github.com/arcalab/team-a

Communication Properties' Characterisation in mCRL2

Receptiveness:

```
[ (r1_finish|c_finish + r2_run + c_start|r1_start|r2_start + r2_finish|c_finish + r1_run)* ](
  ((<c_start> true)  =>  (<c_start|r1_start|r2_start> true)) &&
  ((<r1_finish> true)  =>  (<r1_finish|c_finish> true)) &&
  ((<r2_finish> true)  =>  (<r2_finish|c_finish> true))
)
```

Responsiveness:

```
[ (r1_finish|c_finish + r2_run + c_start|r1_start|r2_start + r2_finish|c_finish + r1_run)* ](
   (<c_finish +
     r1_start|r2_start> true)
   =>
   (<r1_finish|c_finish +
     c_start|r1_start|r2_start +
     r2_finish|c_finish> true)
)
```

Weak Receptiveness:

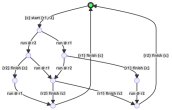
```
[ (r1_finish|c_finish + r2_run + c_start|r1_start|r2_start + r2_finish|c_finish + r1_run)* ](
  ((<r1_finish> true)  =>  (<(r2_run+r2_finish|c_finish)* . r1_finish|c_finish> true)) &&
  ((<r2_finish> true)  =>  (<(r1_finish|c_finish+r1_run)* . r2_finish|c_finish> true)) &&
  ((<c_start> true)  =>  (<(r2_run+r1_run)* . c_start|r1_start|r2_start> true))
)
```

Weak Responsiveness:

```
[ (r1_finish|c_finish + r2_run + c_start|r1_start|r2_start + r2_finish|c_finish + r1_run)* ](
   (<c_finish +
     r1_start|r2_start> true)
   =>
   (<(r2_run+r1_run)* . r1_finish|c_finish +
     c_start|r1_start|r2_start +
     (r2_run+r1_run)* . r2_finish|c_finish> true)
)
```

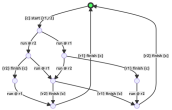
View mCRL2 evidence

Receptiveness: true

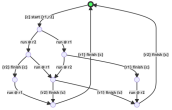


Responsiveness: false

Weak Receptiveness: true



Weak Responsiveness: true

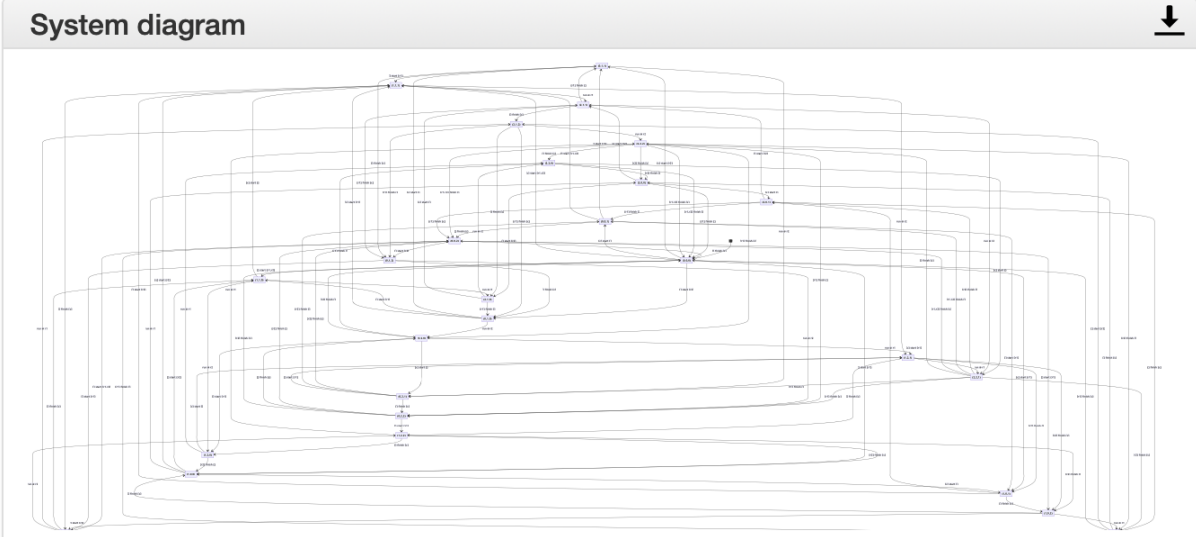


Online prototype: https://github.com/arcalab/team-a

mCRL2 full system:

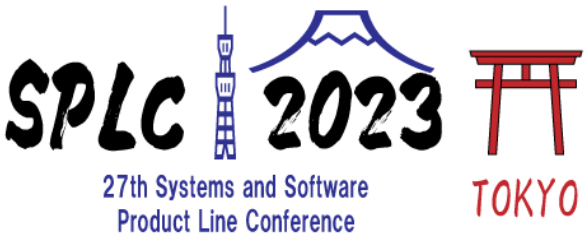
```
act
   r1_start,c_finish,r2_run,c_start,r2_finish,r2_start,r1_finish,r1_run;
proc
   r1(s:Int) =
     (s == 2) -> ( r1_finish.r1(0) ) +
     (s == 1) -> ( r1_run.r1(2) ) +
     (s == 0) -> ( r1_start.r1(1) );
   r2(s:Int) =
     (s == 2) -> ( r2_finish.r2(0) ) +
     (s == 1) -> ( r2_run.r2(2) ) +
     (s == 0) -> ( r2_start.r2(1) );
   c(s:Int) =
     (s == 2) -> ( c_finish.c(0) ) +
     (s == 1) -> ( c_finish.c(2) ) +
     (s == 0) -> ( c_start.c(1) );
init
   allow({
     r1_start,
     c_finish,
     c_start|r2_start,
     c_start,
     c_start|r2_start|r1_start,
     r2_finish|r1_finish,
     c_finish|r2_finish,
     r2_finish,
     r2_start,
     r1_finish,
     c_start|r1_start,
     r1_run,
     c_finish|r2_finish|r1_finish,
     r2_run,
     r2_start|r1_start,
     c_finish|r1_finish},
   r1(0) || r2(0) || c(0));
```

Online prototype: https://github.com/arcalab/team-a



Thanks for your attention! Questions?

I would be glad to meet some of you at SPLC 2023 in Tokyo, Japan



Research and Industry track papers deadline: April 13

More tracks (challenges, demonstrations and tools, journal first, doctoral symposium, hall of fame) plus workshops and tutorials

General chairs: Paolo Arcaini, National Institute of Informatics, Japan
and Maurice ter Beek, ISTI–CNR, Pisa, Italy

https://2023.splc.net/