Featured Team Automata

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

[1]ISTI–CNR, Pisa, Italy

[2]HASLab, INESC TEC & University of Minho, Portugal

[3]Ludwig-Maximilians-Universität München, Munich, Germany

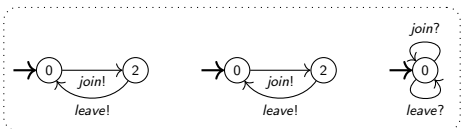
[4]CISTER, ISEP, Polytechnic Institute of Porto, Portugal

T-LADIES kick-off, 6–7 July 2022

Background

Team Automata:[1]

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



Goal: safe communication[3] – *no message loss, no indefinite waiting, ...*

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

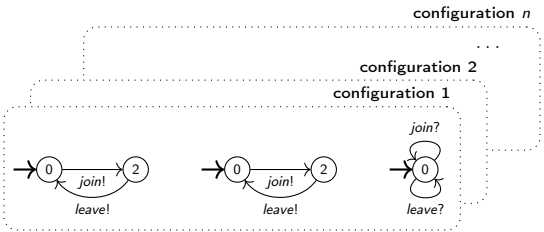
[2] ter Beek, Ellis, Kleijn & Rozenberg, *Synchronizations in team automata for groupware systems*. Comput. Sup. Coop. Work 12, 2003

[3] 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):[4]

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



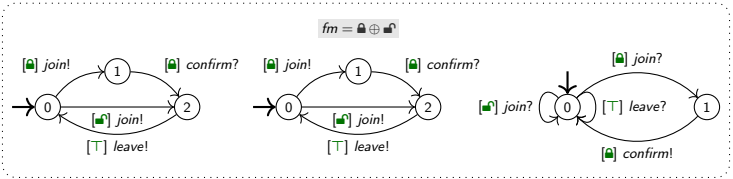
Challenge: system-by-system analysis of safe communication quickly becomes unfeasible

[4] 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:[5]

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

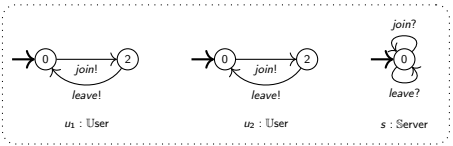


Goal: family-based analysis of safe communication

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

Team Automata

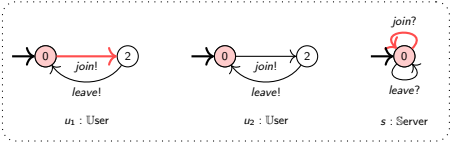
ICTAC'20:



Systems

Team Automata

ICTAC'20:

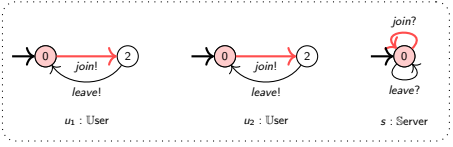


Systems

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

Team Automata

ICTAC'20:

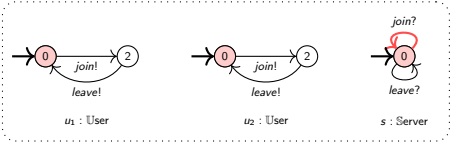


Systems

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

Team Automata

ICTAC'20:



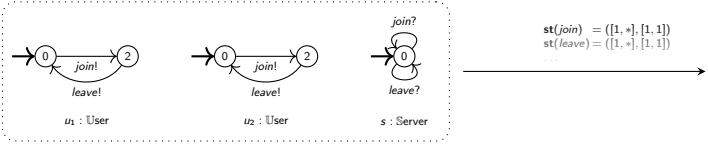
Systems

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

might not be desirable

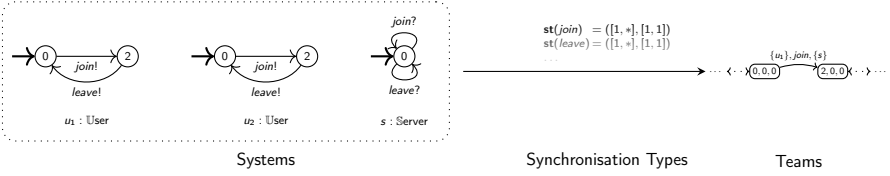
Team Automata

ICTAC'20:



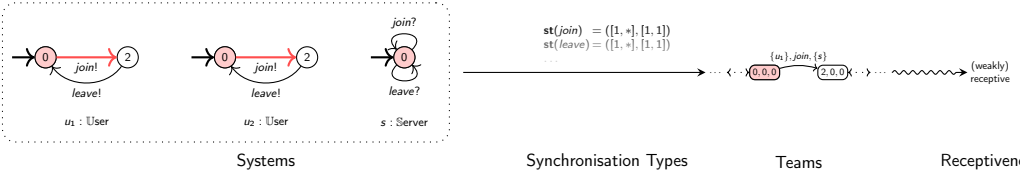
Team Automata

ICTAC'20:



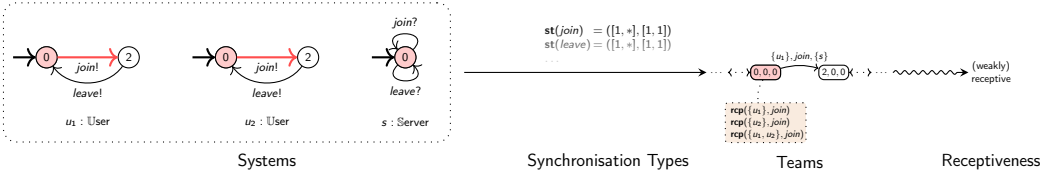
Team Automata

ICTAC'20:

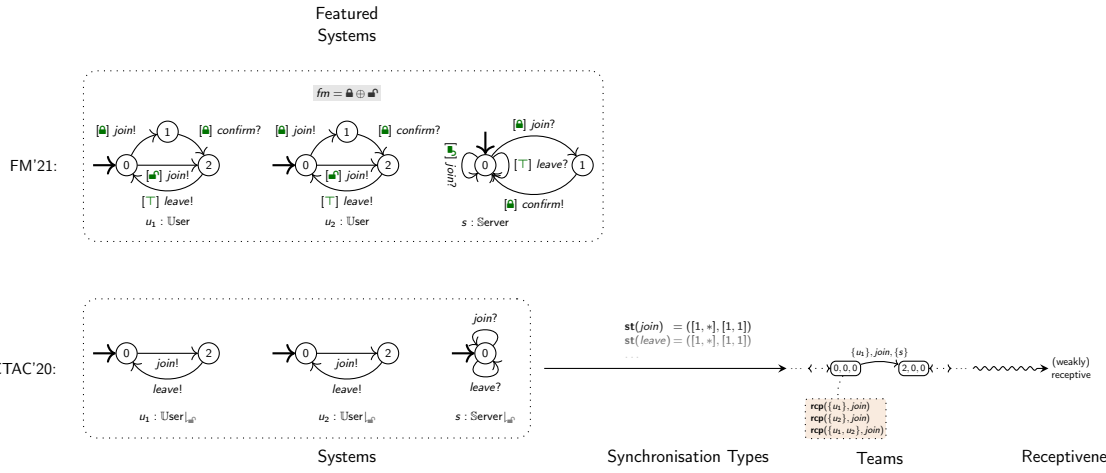


Team Automata

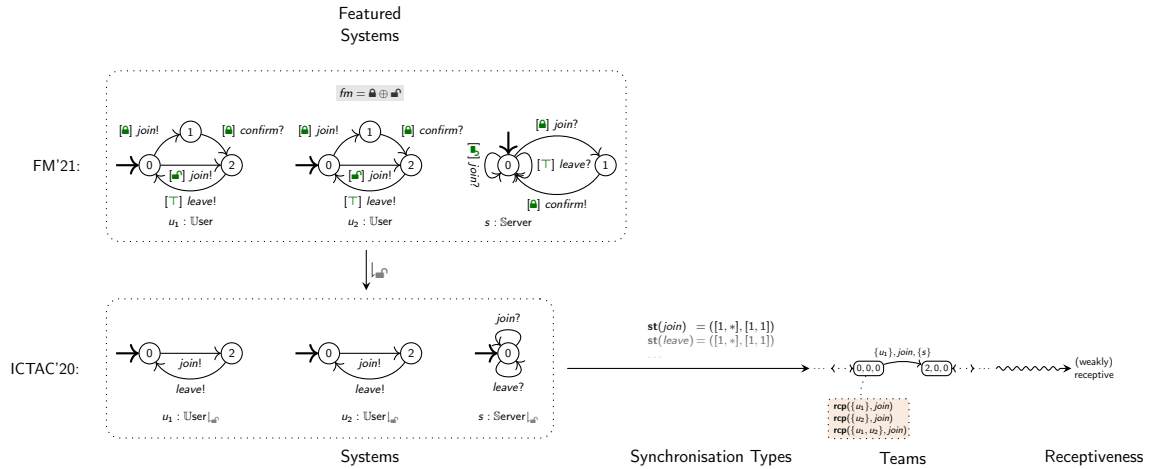
ICTAC'20:



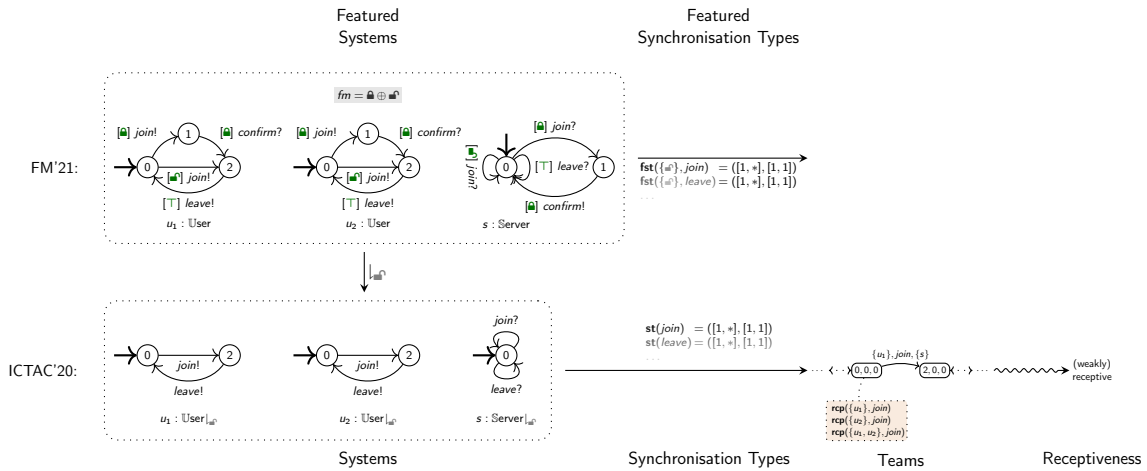
Featured Team Automata



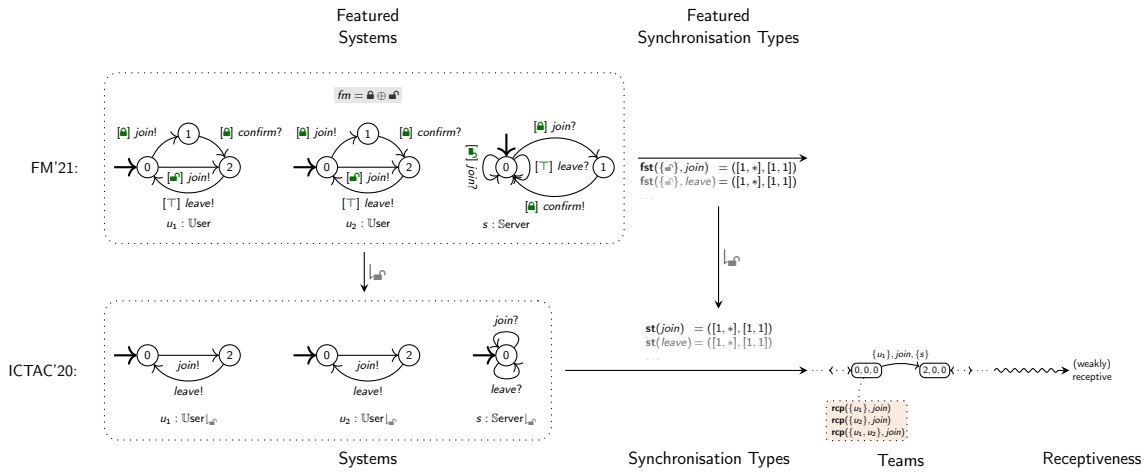
Featured Team Automata



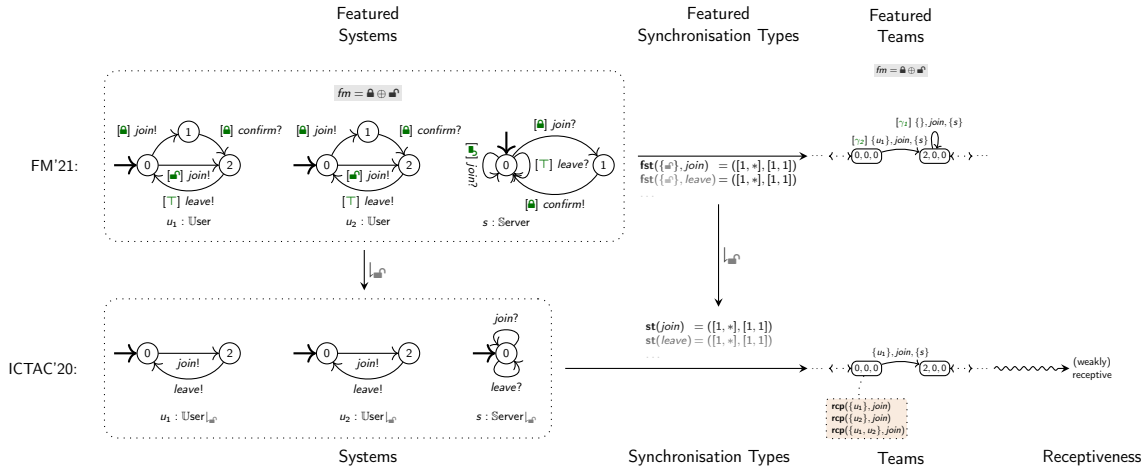
Featured Team Automata



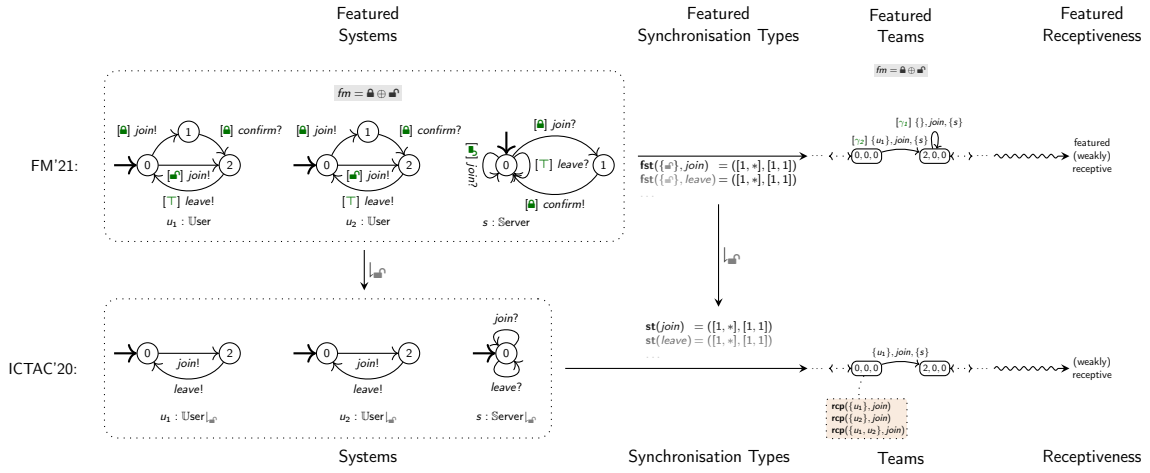
Featured Team Automata



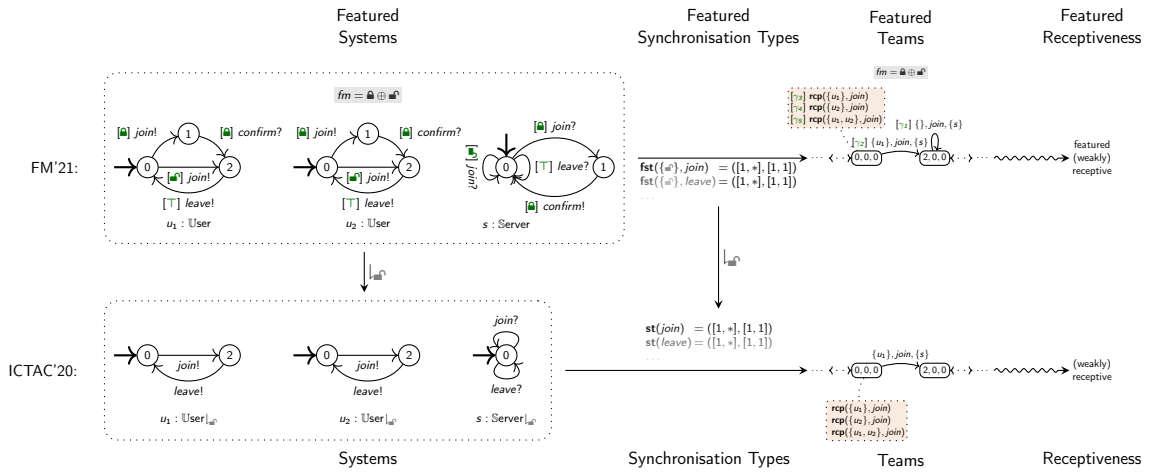
Featured Team Automata



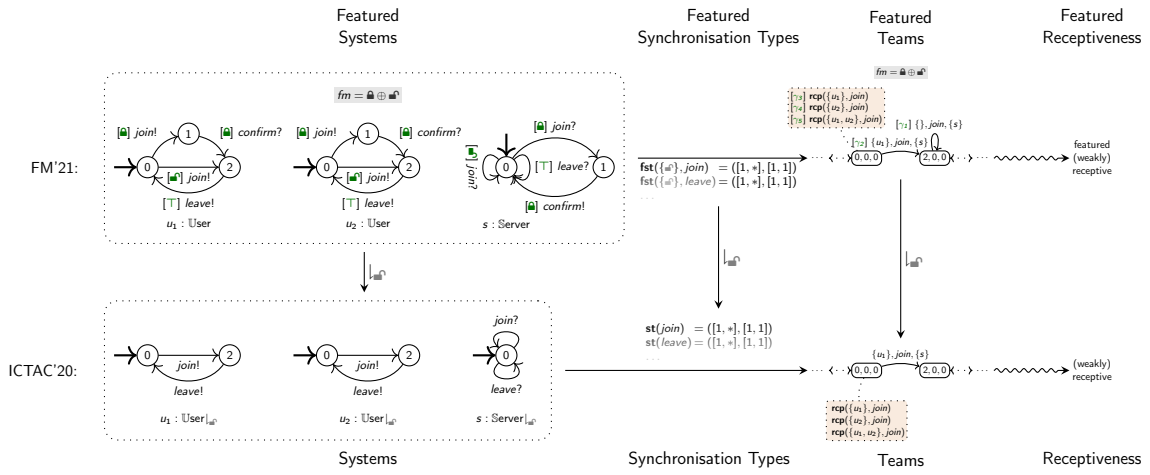
Featured Team Automata



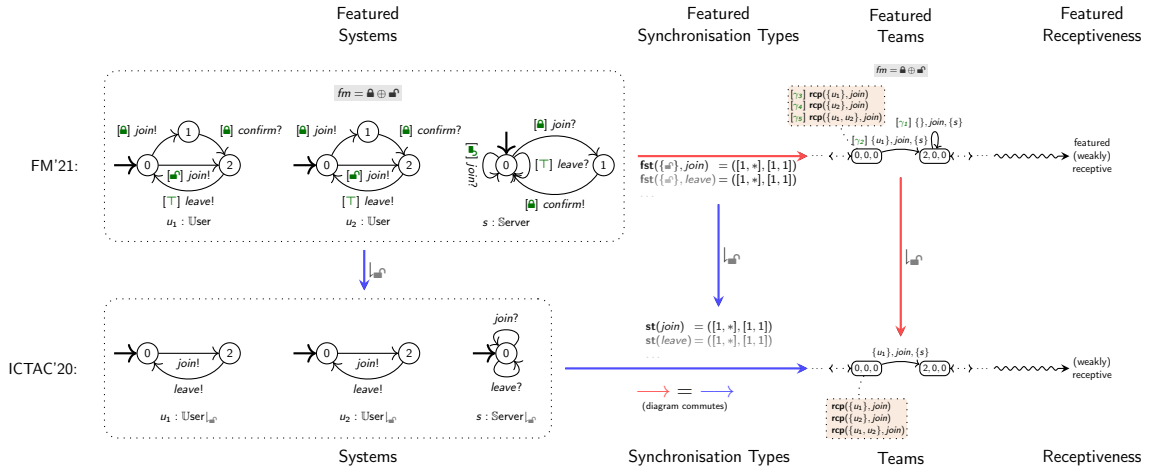
Featured Team Automata



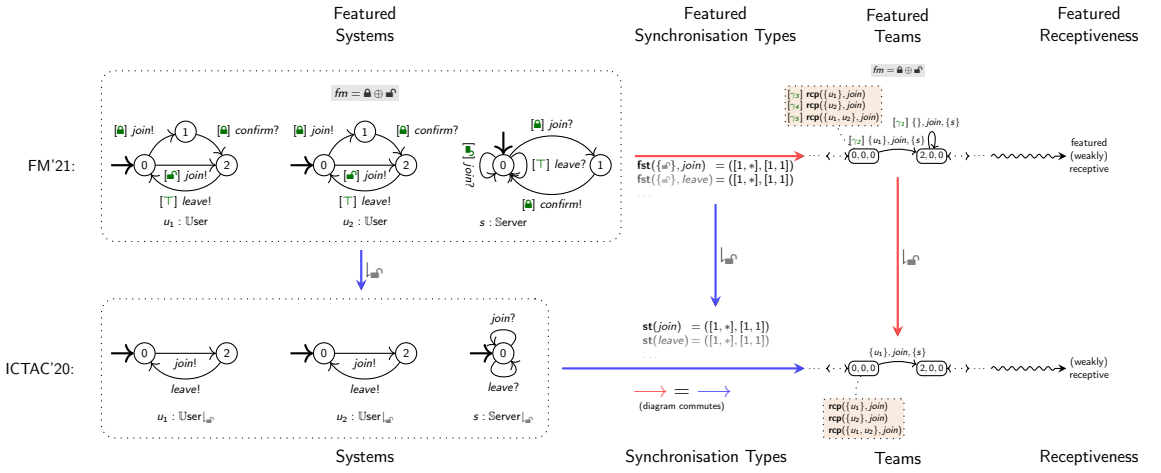
Featured Team Automata



Featured Team Automata

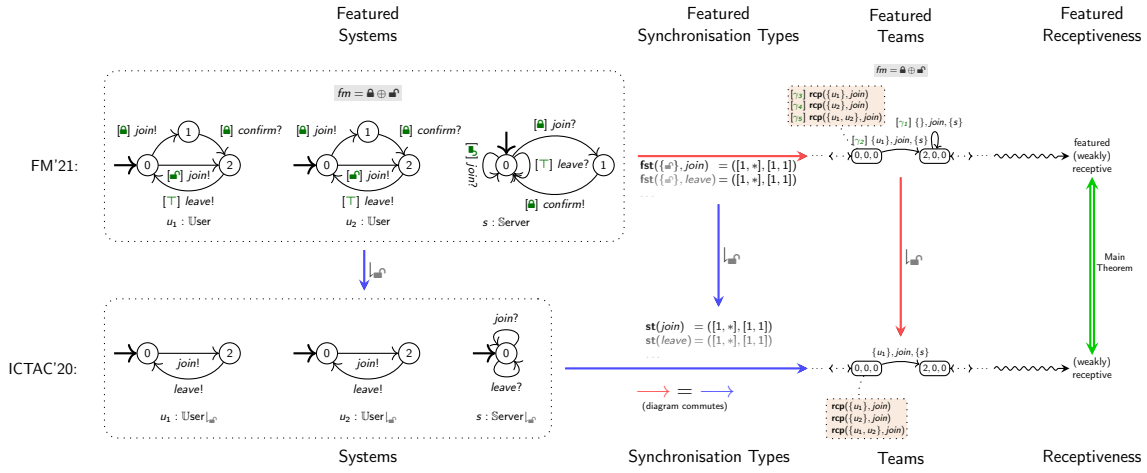


Featured Team Automata



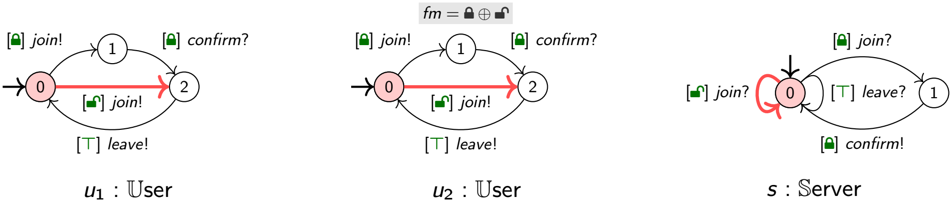
Online prototype: `http://arcatools.org/feta`

Featured Team Automata



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

Featured Team Automata Transitions



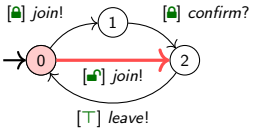
Transitions are constrained with feature expressions by:

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

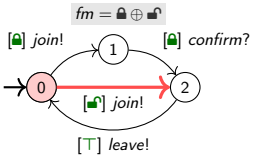
$$\textbf{fst}(\{🔒\}, join) = ([1,1],[1,1]) \qquad \textbf{fst}(\{🔓\}, join) = ([1,*],[1,1])$$

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

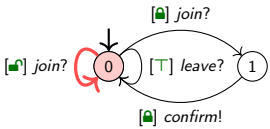
Featured Team Automata Transitions



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



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



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

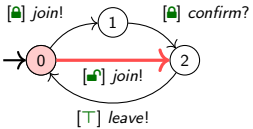
Transitions are constrained with feature expressions by:

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

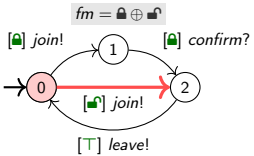
$$\textbf{fst}(\{🔒\}, join) = ([1,1],[1,1]) \qquad \textbf{fst}(\{🔓\}, join) = ([1,*],[1,1])$$

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

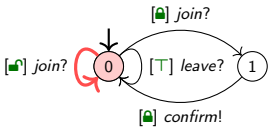
Featured Team Automata Transitions



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



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



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

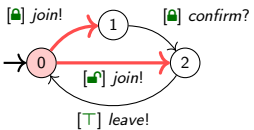
Transitions are constrained with feature expressions by:

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

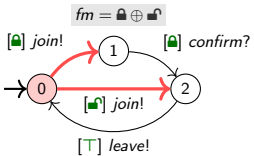
$$\textbf{fst}(\{🔒\}, join) = ([1,1],[1,1]) \qquad \textbf{fst}(\{🔓\}, join) = ([1,*],[1,1])$$

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

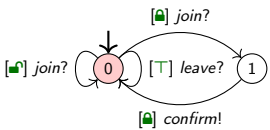
Featured Receptiveness Requirements



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



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



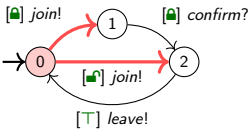
$s : \mathbb{S}\text{erver}$

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

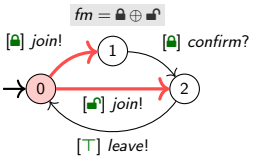
At state $(0,0,0)$:

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

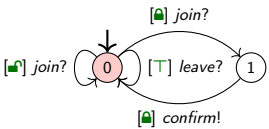
Featured Receptiveness Requirements



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



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



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

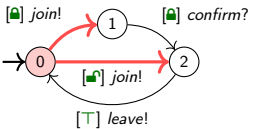
$$\mathbf{fst}(\{🔒\}, join) = ([1,1],[1,1]) \qquad \mathbf{fst}(\{🔓\}, join) = ([1,*],[1,1])$$

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

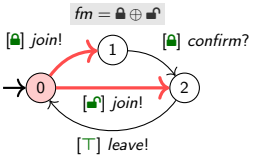
[ ] $\mathbf{rcp}(\{u_1\}, join)$ $\wedge$ [ ] $\mathbf{rcp}(\{u_2\}, join)$ $\wedge$ [ ] $\mathbf{rcp}(\{u_1, u_2\}, join)$

Featured receptiveness requirements are constrained with feature expression by:

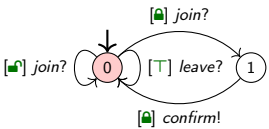
Featured Receptiveness Requirements



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



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



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

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

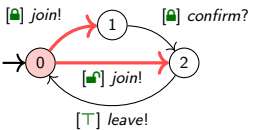
At state $(0,0,0)$:

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

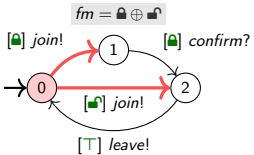
Featured receptiveness requirements are constrained with feature expression by:

- local feature expressions: characterise products with enabled local transitions

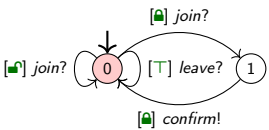
Featured Receptiveness Requirements



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



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



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

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

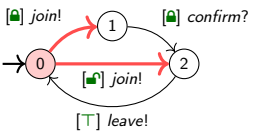
At state $(0,0,0)$:

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

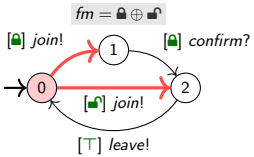
Featured receptiveness requirements are constrained with feature expression by:

- local feature expressions: characterise products with enabled local transitions

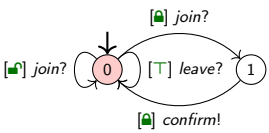
Featured Receptiveness Requirements



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



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



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

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

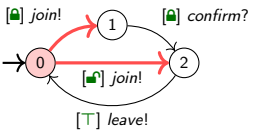
At state $(0,0,0)$:

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

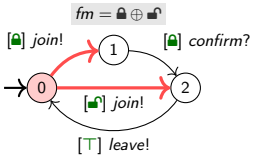
Featured receptiveness requirements are constrained with feature expression by:

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

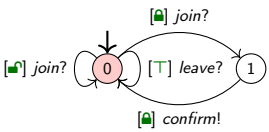
Featured Receptiveness Requirements



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



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



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

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

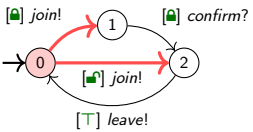
At state $(0,0,0)$:

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

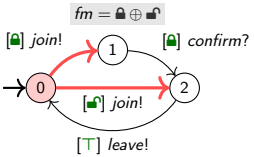
Featured receptiveness requirements are constrained with feature expression by:

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

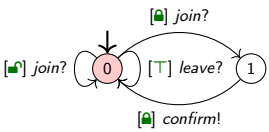
Featured Receptiveness Requirements



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



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



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

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

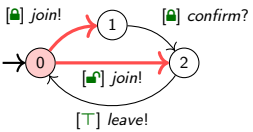
At state $(0,0,0)$:

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

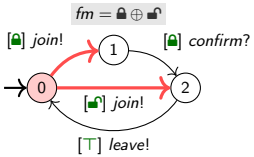
Featured receptiveness requirements are constrained with feature expression by:

- 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

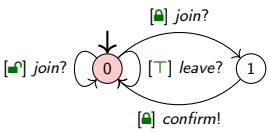
Featured Receptiveness Requirements



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



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



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

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

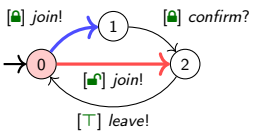
At state $(0,0,0)$:

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

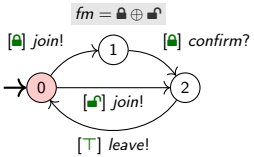
Featured receptiveness requirements are constrained with feature expression by:

- 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

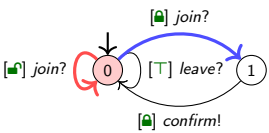
Compliance with requirements



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



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



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

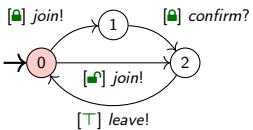
At state $(0,0,0)$:

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

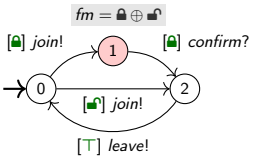
$$\{🔒\} : \ (0,0,0) \xrightarrow{[🔒 \wedge fm]\ (\{u_1\}, join, \{s\})} \mathbf{fst}[\mathcal{S}](1,0,1)$$

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

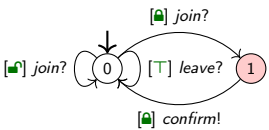
Compliance with requirements



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



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

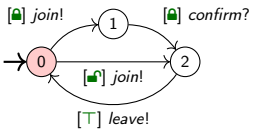


$s : \mathbb{S}\text{erver}$

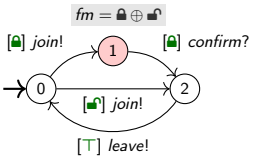
At state $(0, 1, 1)$:

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

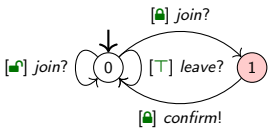
Compliance with requirements



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



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

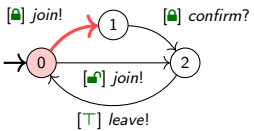


$s : \mathbb{S}\text{erver}$

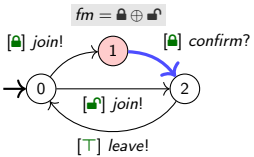
At state $(0, 1, 1)$:

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

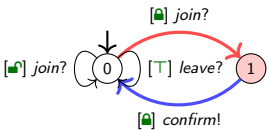
Weak compliance with requirements



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



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



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

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

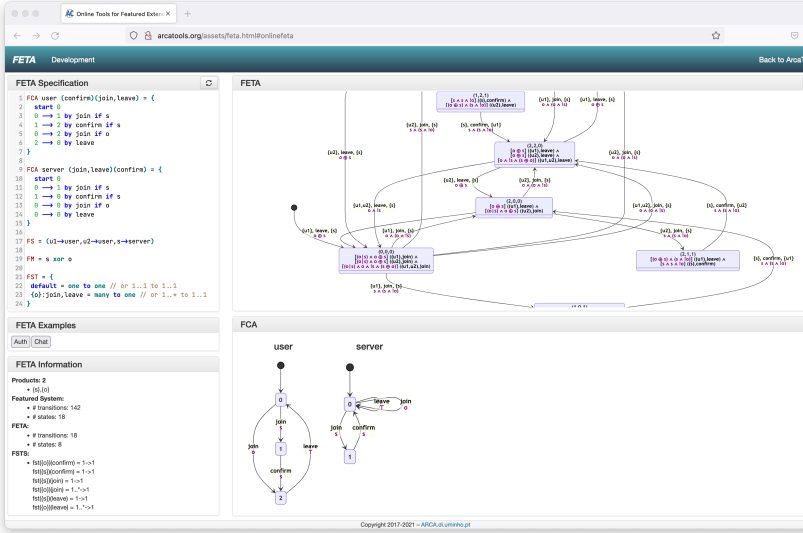
$$[🔒 \wedge \neg 🔓]\ \mathbf{rcp}(\{u_1\}, join)\ \wedge\ \ldots$$

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

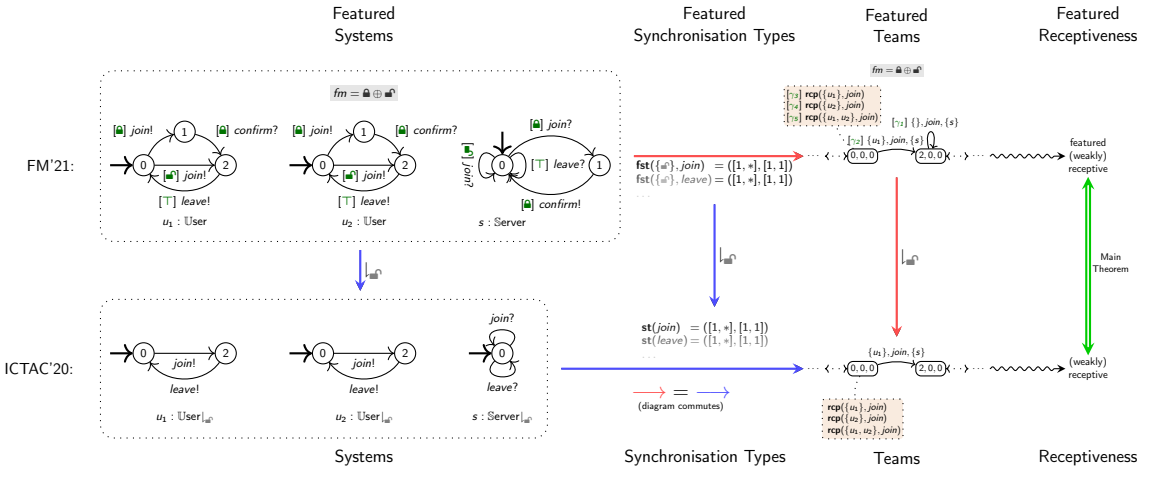
Online prototype

- Specify
- Generate*
- Visualise
- Statistics

*SAT solver to solve *fm*

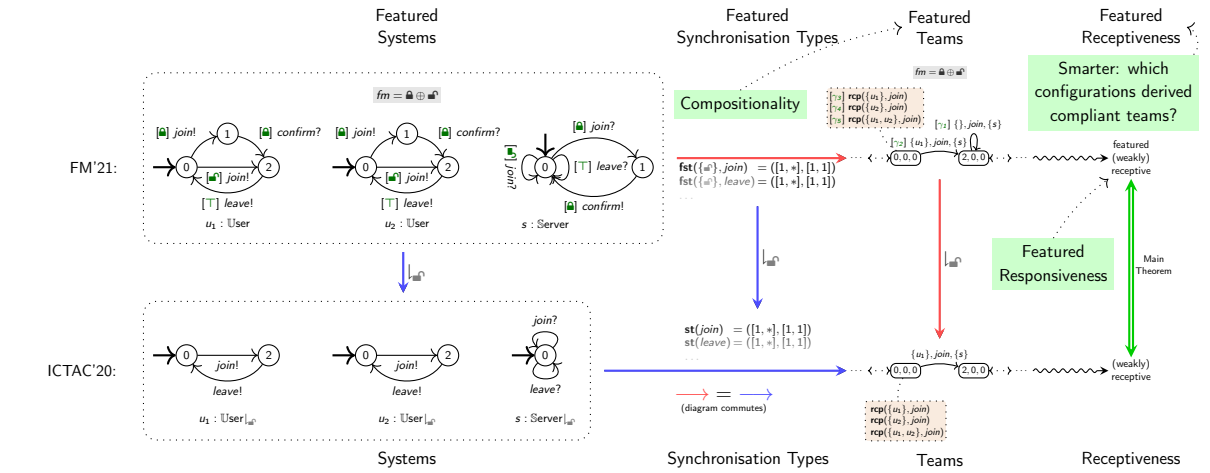


Wrapping up



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

Thank you for your attention!
Questions?

Formal Methods and Tools Lab