



Featured Team Automata

Maurice H. ter Beek¹ Guillermina Cledou² Rolf Hennicker³ José Proença⁴

¹ISTI-CNR, Pisa, Italy

²HASLab, INESC TEC & University of Minho, Portugal

³Ludwig-Maximilians-Universität München, Munich, Germany

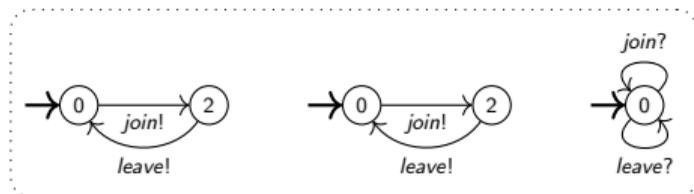
⁴CISTER, ISEP, Polytechnic Institute of Porto, Portugal

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

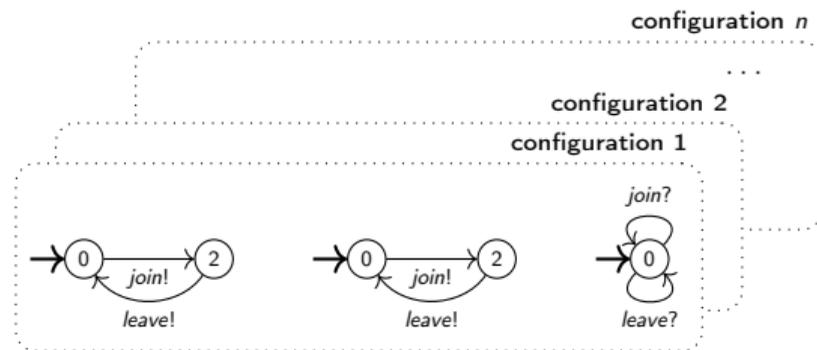
² ter Beek, Ellis, Kleijn & Rozenberg, *Synchronizations in team automata for groupware systems*. Comput. Sup. Coop. Work 12, 2003

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



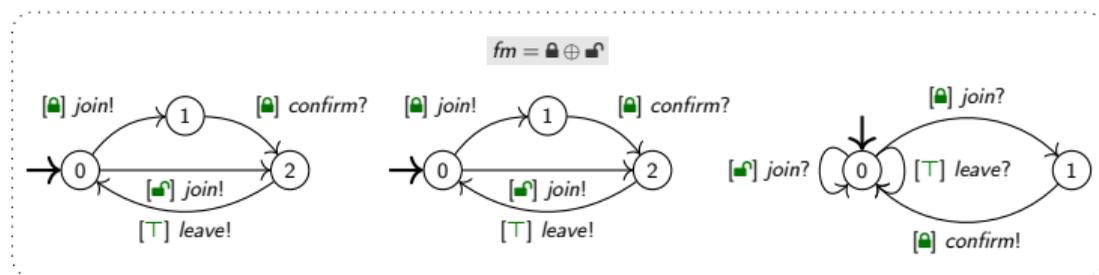
Challenge: system-by-system analysis of safe communication quickly becomes **unfeasible**

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.: lock , unlock), and a **feature model** ($\text{lock} \oplus \text{unlock}$)

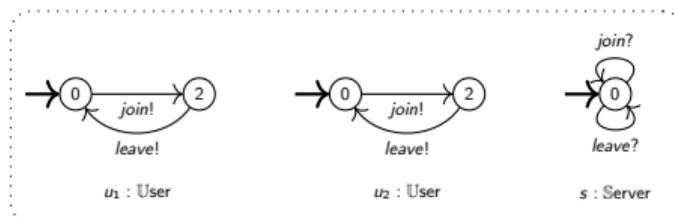


Goal: **family-based analysis** of safe communication

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

Team Automata

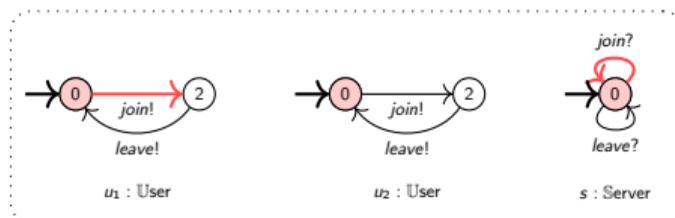
ICTAC'20:



Systems

Team Automata

ICTAC'20:

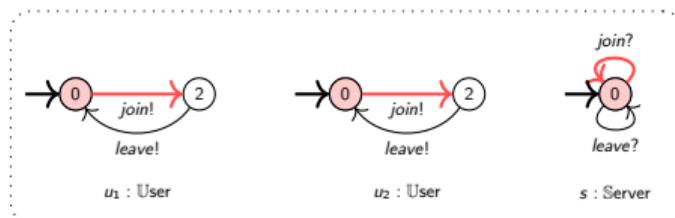


Systems

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

Team Automata

ICTAC'20:

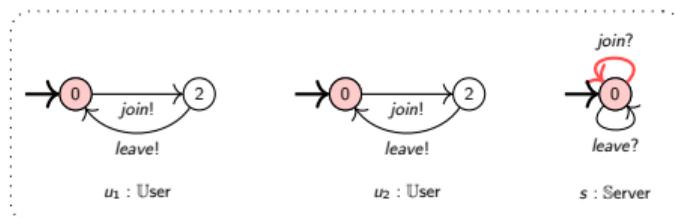


Systems

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

Team Automata

ICTAC'20:

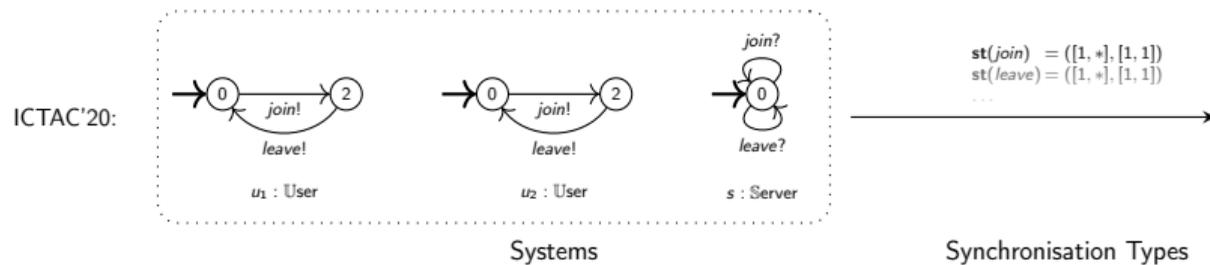


Systems

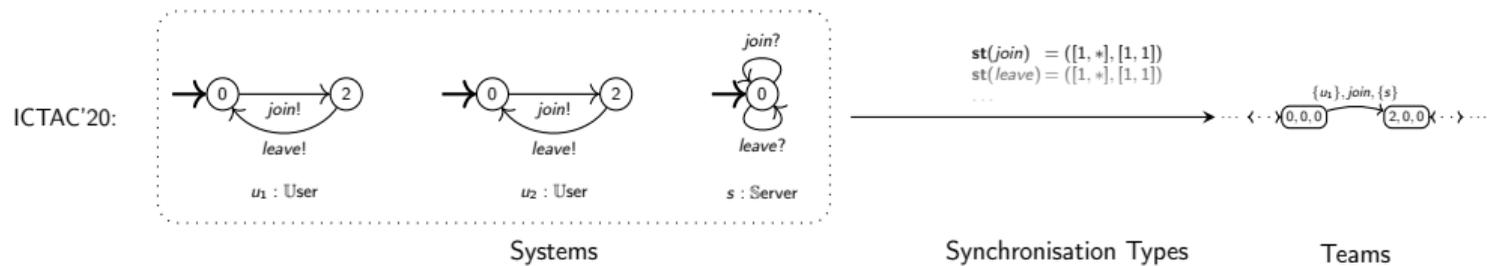
$$(0, 0, 0) \xrightarrow{(\{\}, \text{join}, \{s\})} (0, 0, 0)$$

might not be desirable

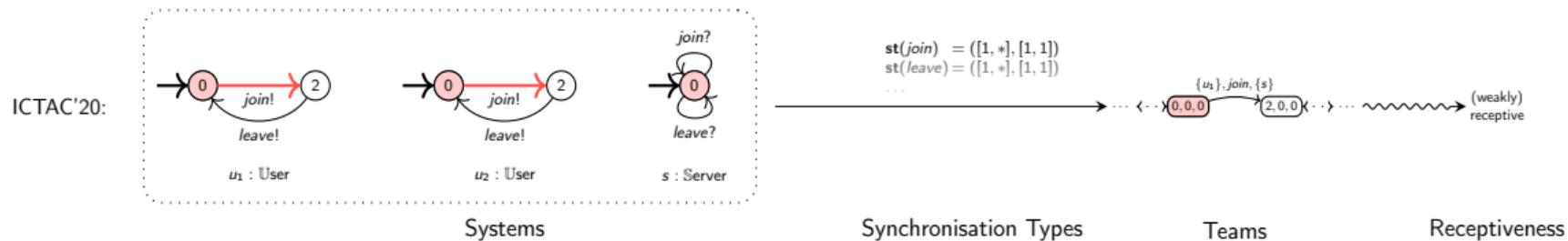
Team Automata



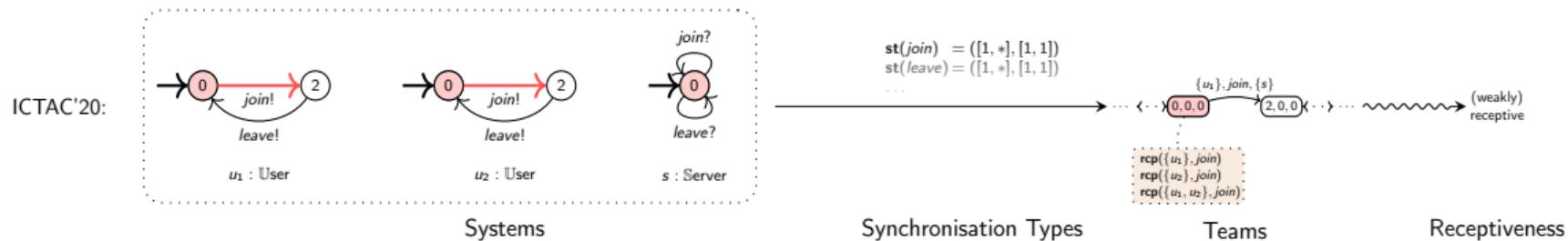
Team Automata



Team Automata

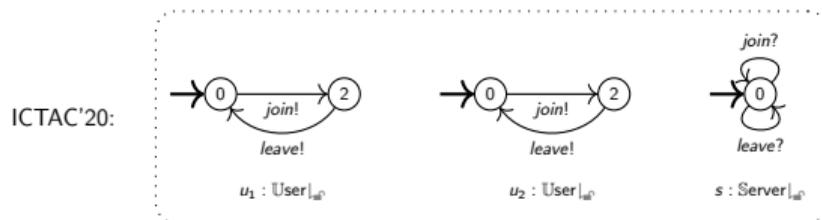
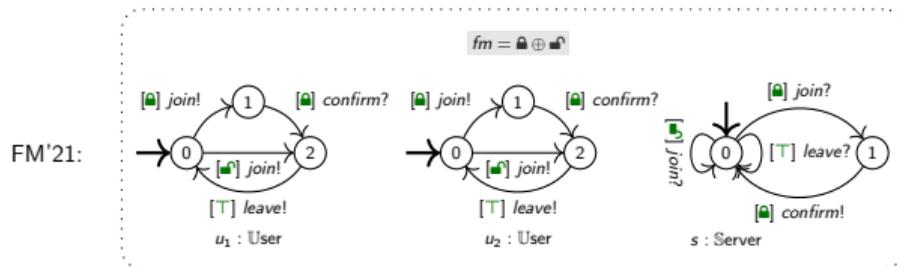


Team Automata



Featured Team Automata

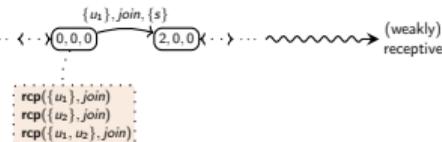
Featured Systems



Systems

$st(\text{join}) = ([1, *], [1, 1])$
 $st(\text{leave}) = ([1, *], [1, 1])$

Synchronisation Types

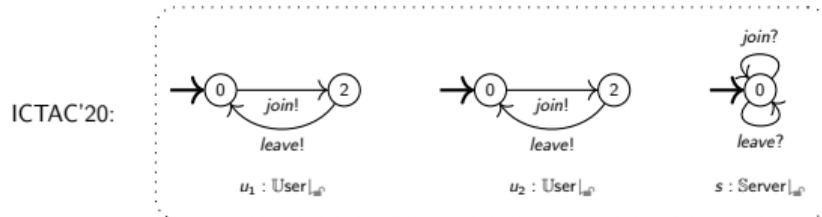
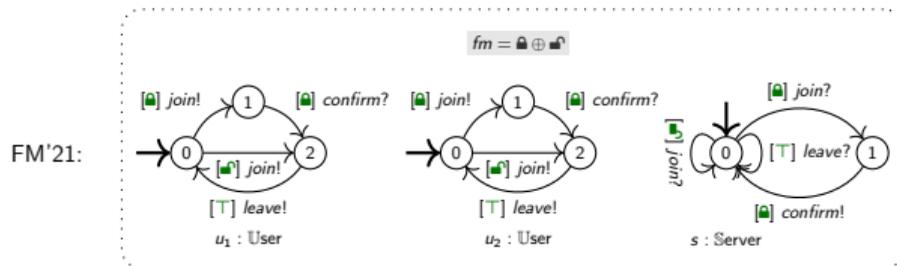


Teams

Receptiveness

Featured Team Automata

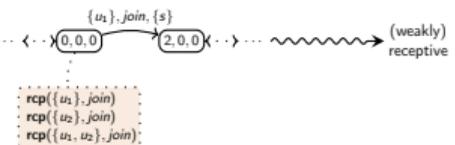
Featured Systems



Systems

$st(join) = ([1, *], [1, 1])$
 $st(leave) = ([1, *], [1, 1])$

Synchronisation Types



Teams

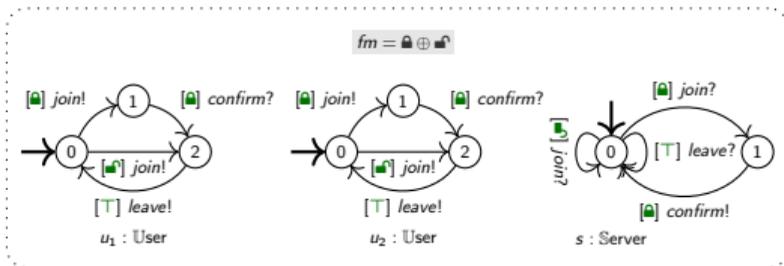
Receptiveness

Featured Team Automata

Featured Systems

Featured Synchronisation Types

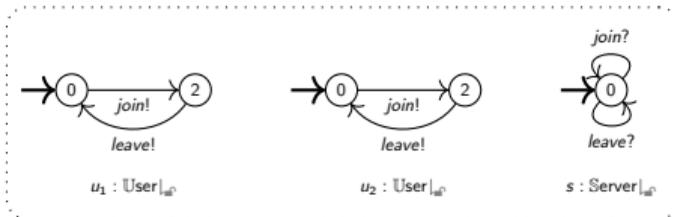
FM'21:



fst($\{a\}, join$) = $([1, *], [1, 1])$
 fst($\{a\}, leave$) = $([1, *], [1, 1])$
 ...



ICTAC'20:

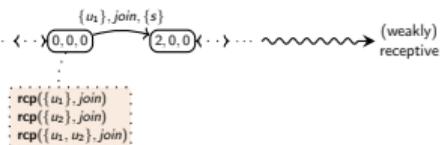


st(join) = $([1, *], [1, 1])$
 st(leave) = $([1, *], [1, 1])$
 ...

Synchronisation Types

Teams

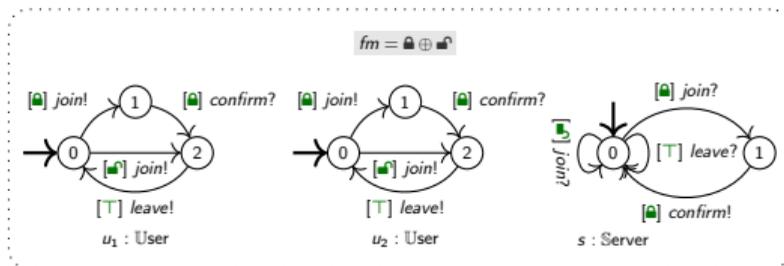
Receptiveness



Featured Team Automata

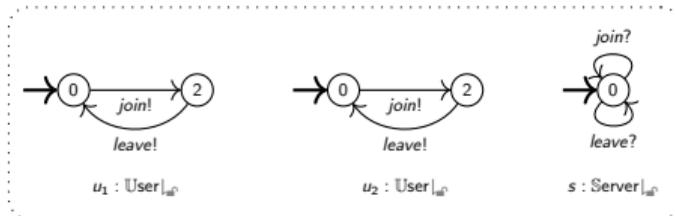
Featured Systems

FM'21:



Systems

ICTAC'20:



Featured Synchronisation Types

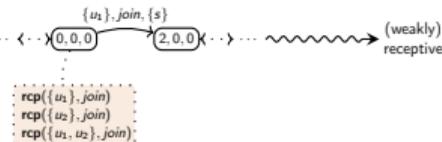
$$\begin{aligned} \text{fst}(\{a\}, \text{join}) &= ([1, *], [1, 1]) \\ \text{fst}(\{a\}, \text{leave}) &= ([1, *], [1, 1]) \\ \dots \end{aligned}$$

$$\begin{aligned} \text{st}(\text{join}) &= ([1, *], [1, 1]) \\ \text{st}(\text{leave}) &= ([1, *], [1, 1]) \\ \dots \end{aligned}$$

Synchronisation Types

Teams

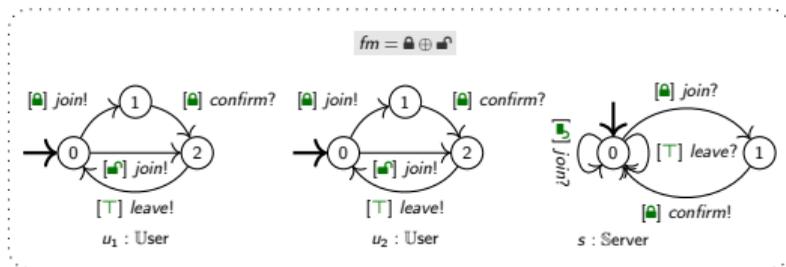
Receptiveness



Featured Team Automata

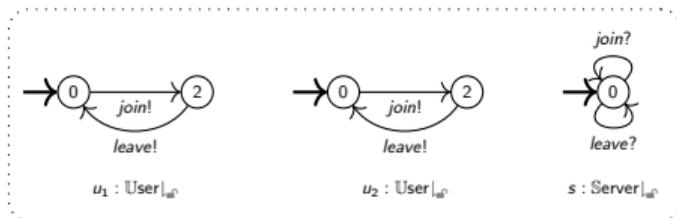
Featured Systems

FM'21:



Systems

ICTAC'20:



Featured Synchronisation Types

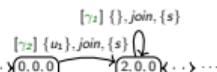
$\text{fst}(\{u_1\}, \text{join}) = ([1, *], [1, 1])$
 $\text{fst}(\{u_1\}, \text{leave}) = ([1, *], [1, 1])$

$\text{st}(\text{join}) = ([1, *], [1, 1])$
 $\text{st}(\text{leave}) = ([1, *], [1, 1])$

Synchronisation Types

Featured Teams

$fm = u_1 \oplus u_2 \oplus s$



$\{u_1\}, \text{join}, \{s\}$
 $\{u_2\}, \text{join}, \{s\}$
 $\text{rcp}(\{u_1\}, \text{join})$
 $\text{rcp}(\{u_2\}, \text{join})$
 $\text{rcp}(\{u_1, u_2\}, \text{join})$

Teams

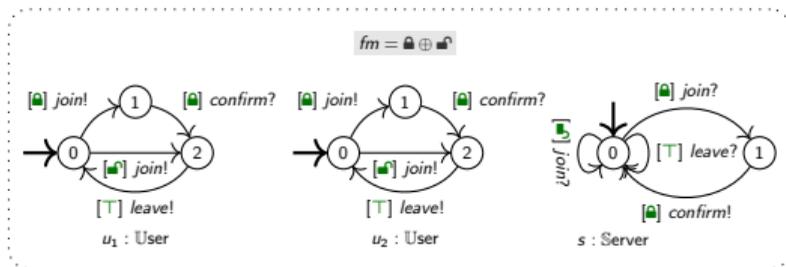
Receptiveness

(weakly) receptive

Featured Team Automata

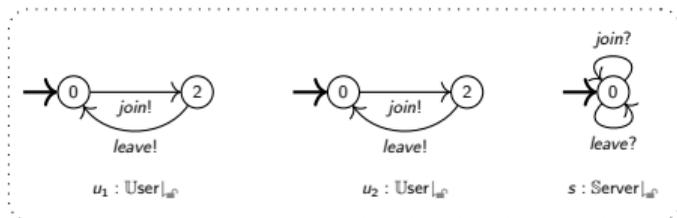
Featured Systems

FM'21:



Systems

ICTAC'20:



Featured Synchronisation Types

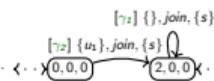
fst($\{u\}$, join) = ([1, *], [1, 1])
fst($\{u\}$, leave) = ([1, *], [1, 1])

st(join) = ([1, *], [1, 1])
st(leave) = ([1, *], [1, 1])

Synchronisation Types

Featured Teams

$fm = \text{lock} \oplus \text{join}$



$\{u_1\}.join, \{s\}$
rcp($\{u_1\}.join$)
rcp($\{u_2\}.join$)
rcp($\{u_1, u_2\}.join$)

Teams

Featured Receptiveness

featured (weakly) receptive

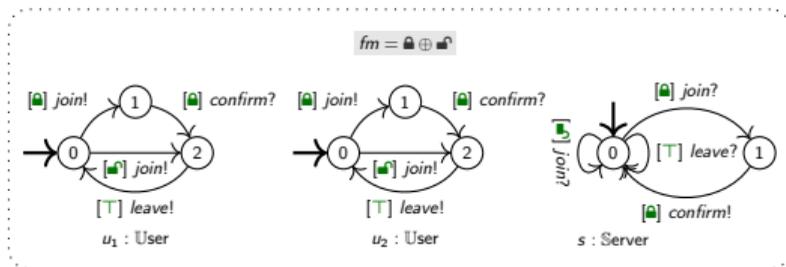
(weakly) receptive

Receptiveness

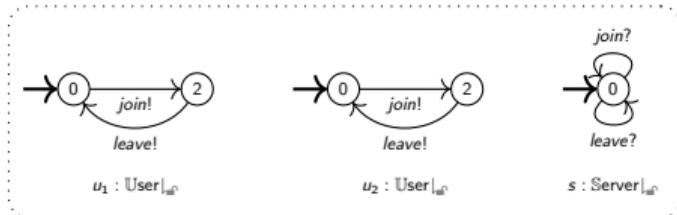
Featured Team Automata

Featured Systems

FM'21:



ICTAC'20:



Systems

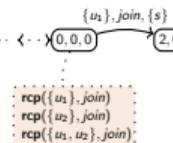
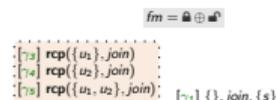
Featured Synchronisation Types

fst($\{a\}, join$) = $([1, *], [1, 1])$
 fst($\{a\}, leave$) = $([1, *], [1, 1])$

st(join) = $([1, *], [1, 1])$
 st(leave) = $([1, *], [1, 1])$

Synchronisation Types

Featured Teams



Teams

Featured Receptiveness

featured (weakly) receptive

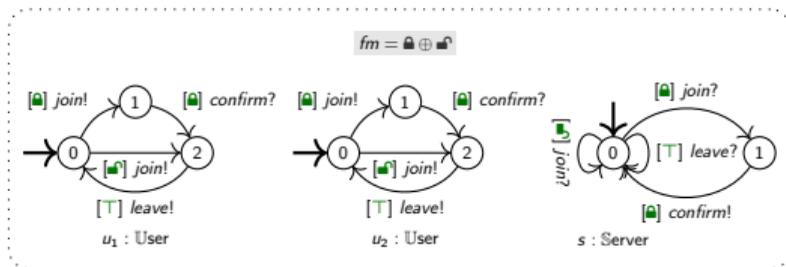
(weakly) receptive

Receptiveness

Featured Team Automata

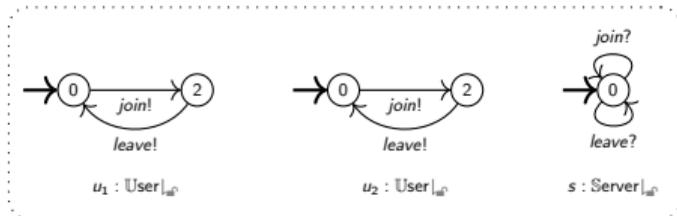
Featured Systems

FM'21:



Systems

ICTAC'20:



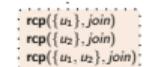
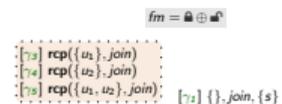
Featured Synchronisation Types

fst($\{a\}, join$) = $([1, *], [1, 1])$
 fst($\{a\}, leave$) = $([1, *], [1, 1])$

st(join) = $([1, *], [1, 1])$
 st(leave) = $([1, *], [1, 1])$

Synchronisation Types

Featured Teams



Teams

Featured Receptiveness

featured (weakly) receptive

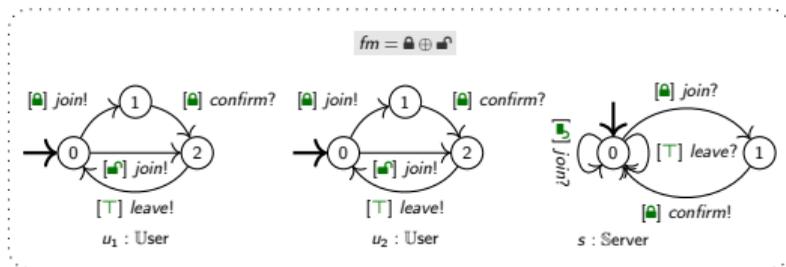
(weakly) receptive

Receptiveness

Featured Team Automata

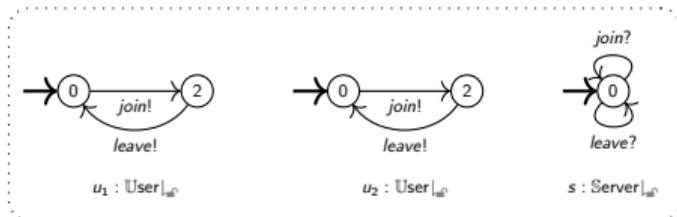
Featured Systems

FM'21:



Systems

ICTAC'20:



Featured Synchronisation Types

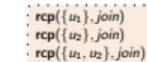
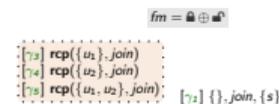
$\text{fst}(\{a\}, \text{join}) = ([1, *], [1, 1])$
 $\text{fst}(\{a\}, \text{leave}) = ([1, *], [1, 1])$

$\text{st}(\text{join}) = ([1, *], [1, 1])$
 $\text{st}(\text{leave}) = ([1, *], [1, 1])$

$\rightarrow = \rightarrow$
 (diagram commutes)

Synchronisation Types

Featured Teams



Teams

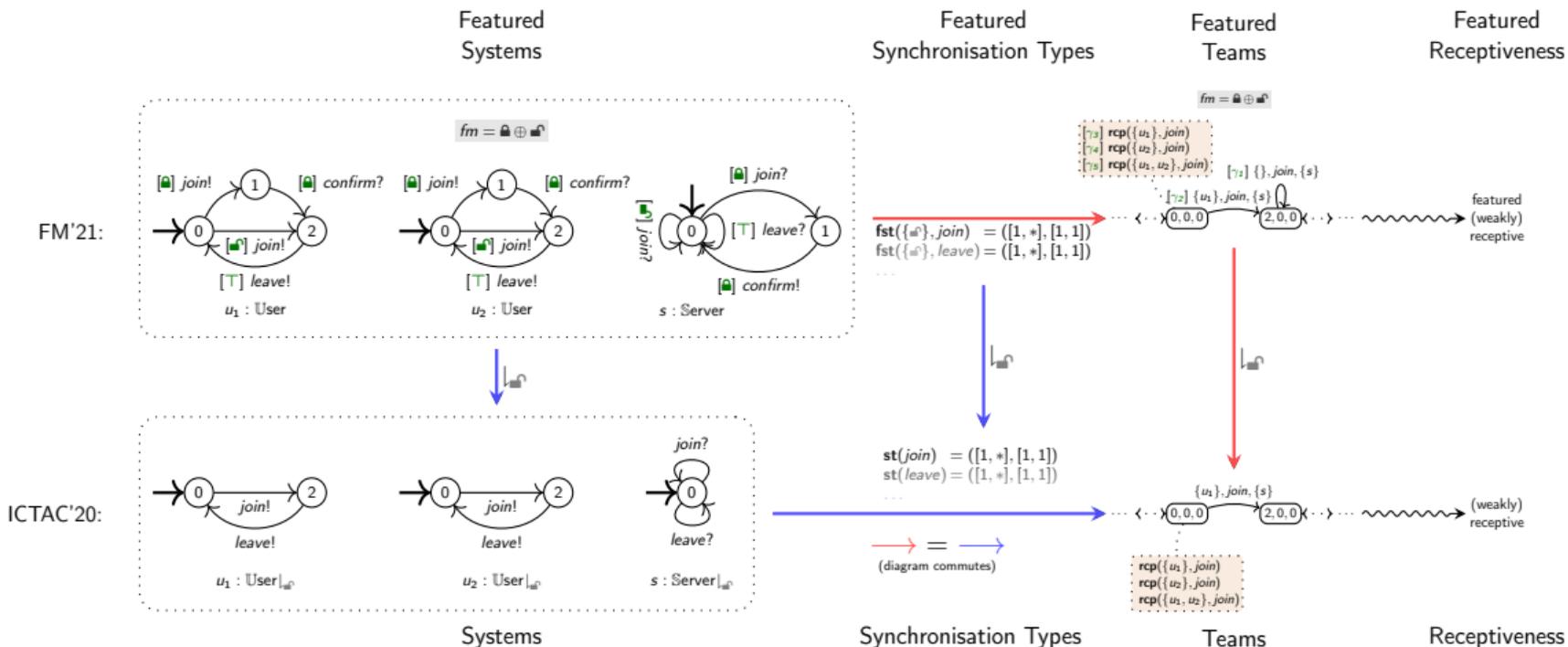
Featured Receptiveness

featured (weakly) receptive

(weakly) receptive

Receptiveness

Featured Team Automata

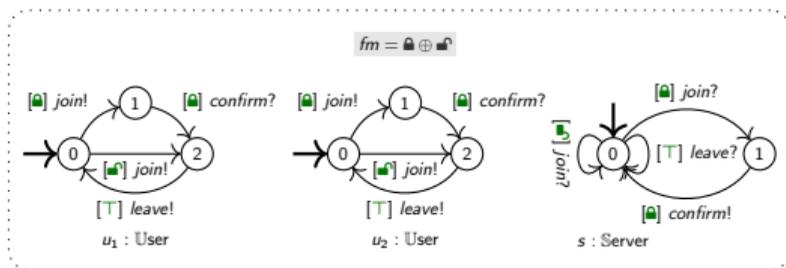


Online prototype: <http://arcatools.org/feta>

Featured Team Automata

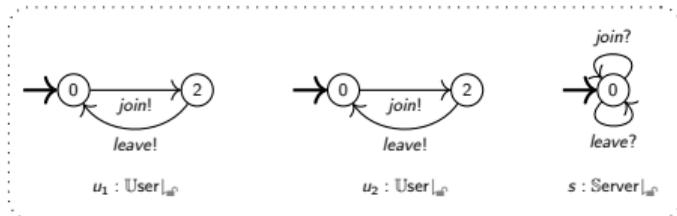
Featured Systems

FM'21:



Systems

ICTAC'20:



Featured Synchronisation Types

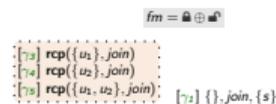
fst($\{a\}, join$) = $([1, *], [1, 1])$
 fst($\{a\}, leave$) = $([1, *], [1, 1])$

st(join) = $([1, *], [1, 1])$
 st(leave) = $([1, *], [1, 1])$

$\rightarrow = \rightarrow$
 (diagram commutes)

Synchronisation Types

Featured Teams



$[72] \{u_1\}.join, \{s\}$

$\{u_1\}.join, \{s\}$



Teams

Featured Receptiveness

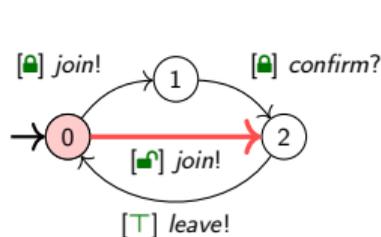
featured (weakly) receptive

(weakly) receptive

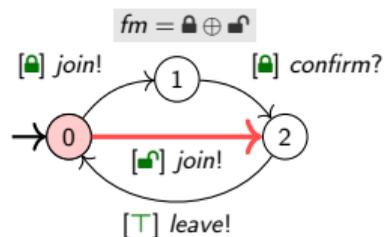
Main Theorem

Online prototype: <http://arcatools.org/feta>

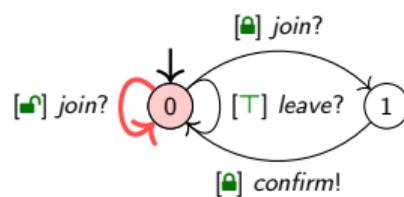
Featured Team Automata Transitions



$u_1 : \text{User}$



$u_2 : \text{User}$



$s : \text{Server}$

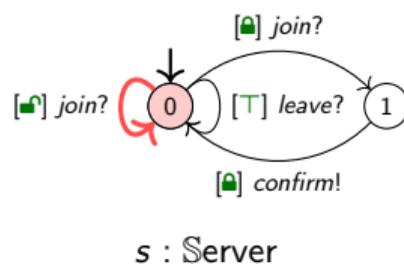
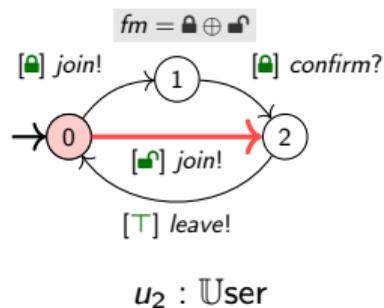
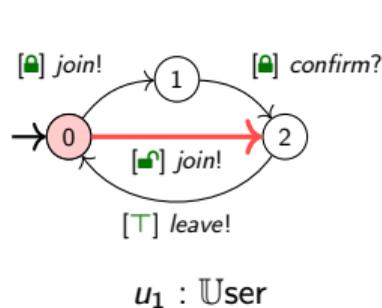
Transitions are **constrained** with feature expressions by:

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

$$\text{fst}(\{\text{lock}\}, \text{join}) = ([1, 1], [1, 1]) \quad \text{fst}(\{\text{unlock}\}, \text{join}) = ([1, *], [1, 1])$$

$$(0, 0, 0) \xrightarrow{[\quad](\{u_1, u_2\}, \text{join}, \{s\})} \text{fst}[S] (2, 2, 0)$$

Featured Team Automata Transitions



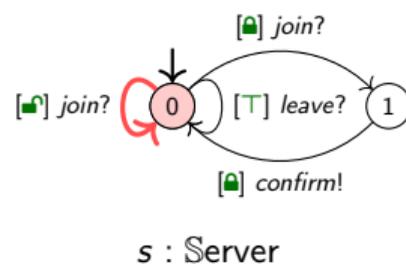
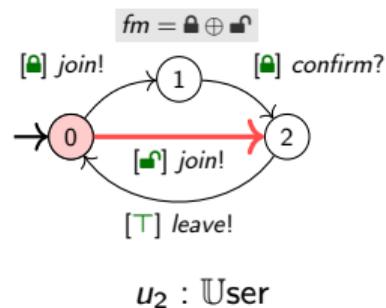
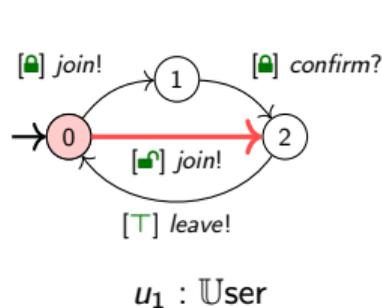
Transitions are **constrained** with feature expressions by:

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

$$\text{fst}(\{\text{lock}\}, \text{join}) = ([1, 1], [1, 1]) \quad \text{fst}(\{\text{unlock}\}, \text{join}) = ([1, *], [1, 1])$$

$$(0, 0, 0) \xrightarrow{[\text{lock} \wedge \text{lock} \wedge \text{lock}]} [(\{u_1, u_2\}, \text{join}, \{s\})]_{\text{fst}[S]} (2, 2, 0)$$

Featured Team Automata Transitions



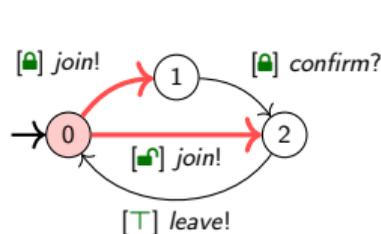
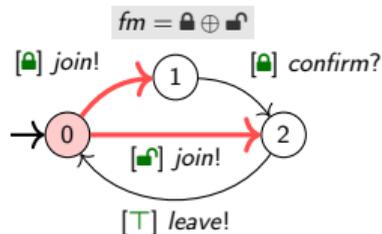
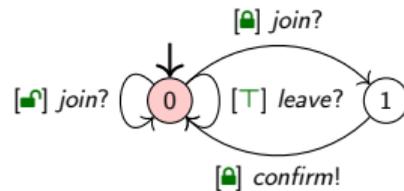
Transitions are **constrained** with feature expressions by:

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

$$\text{fst}(\{\text{lock}\}, \text{join}) = ([1, 1], [1, 1]) \quad \text{fst}(\{\text{unlock}\}, \text{join}) = ([1, *], [1, 1])$$

$$(0, 0, 0) \xrightarrow{[\text{lock} \wedge \text{lock} \wedge \text{lock} \wedge \text{lock} \wedge \text{lock} \wedge \neg \text{lock}]}_{\text{fst}[S]} (\{u_1, u_2\}, \text{join}, \{s\}) (2, 2, 0)$$

Featured Receptiveness Requirements

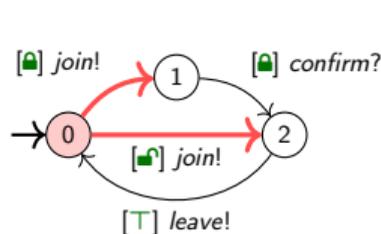
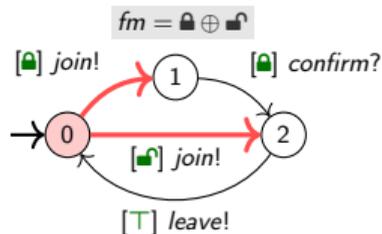
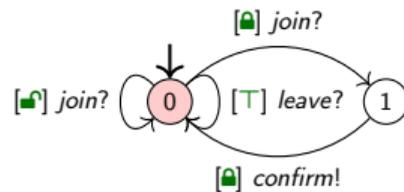
 $u_1 : \text{User}$  $u_2 : \text{User}$  $s : \text{Server}$

$$\text{fst}(\{\text{lock}\}, \text{join}) = ([1, 1], [1, 1]) \quad \text{fst}(\{\text{key}\}, \text{join}) = ([1, *], [1, 1])$$

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

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

Featured Receptiveness Requirements

 $u_1 : \text{User}$  $u_2 : \text{User}$  $s : \text{Server}$

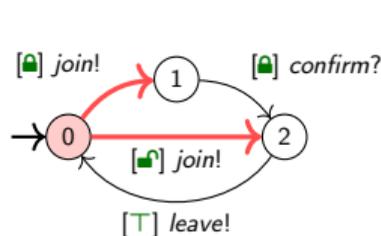
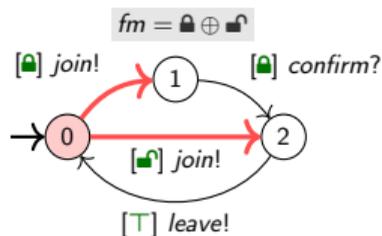
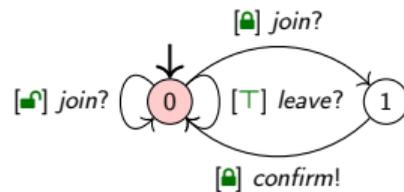
$$\text{fst}(\{\text{lock}\}, \text{join}) = ([1, 1], [1, 1]) \quad \text{fst}(\{\text{lock} \oplus \text{lock}\}, \text{join}) = ([1, *], [1, 1])$$

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

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

Featured receptiveness requirements are **constrained** with feature expression by:

Featured Receptiveness Requirements

 $u_1 : \text{User}$  $u_2 : \text{User}$  $s : \text{Server}$

$$\text{fst}(\{\text{lock}\}, \text{join}) = ([1, 1], [1, 1]) \quad \text{fst}(\{\text{lock}\}, \text{join}) = ([1, *], [1, 1])$$

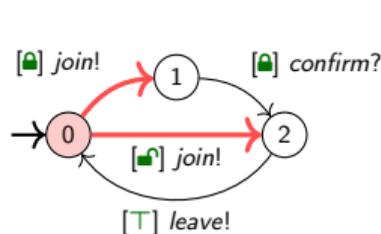
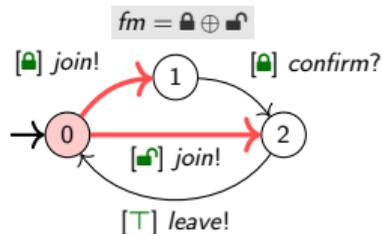
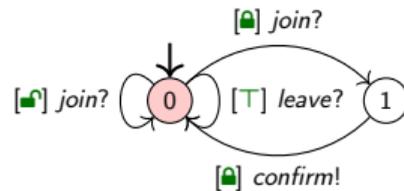
At state $(0, 0, 0)$:

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

Featured receptiveness requirements are **constrained** with feature expression by:

- **local feature expressions**: characterise products with enabled local transitions

Featured Receptiveness Requirements

 $u_1 : \text{User}$  $u_2 : \text{User}$  $s : \text{Server}$

$$\text{fst}(\{\text{lock}\}, \text{join}) = ([1, 1], [1, 1]) \quad \text{fst}(\{\text{lock}\}, \text{join}) = ([1, *], [1, 1])$$

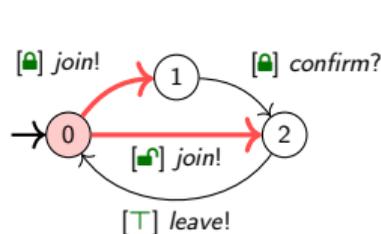
At state $(0, 0, 0)$:

$$[\text{lock} \vee \text{lock}] \text{rcp}(\{u_1\}, \text{join}) \wedge [] \text{rcp}(\{u_2\}, \text{join}) \wedge [] \text{rcp}(\{u_1, u_2\}, \text{join})$$

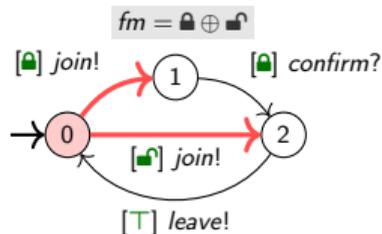
Featured receptiveness requirements are **constrained** with feature expression by:

- **local feature expressions**: characterise products with enabled local transitions

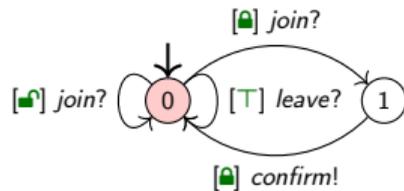
Featured Receptiveness Requirements



$u_1 : \text{User}$



$u_2 : \text{User}$



$s : \text{Server}$

$$\text{fst}(\{\text{lock}\}, \text{join}) = ([1, 1], [1, 1]) \quad \text{fst}(\{\text{lock} \oplus \text{lock}\}, \text{join}) = ([1, *], [1, 1])$$

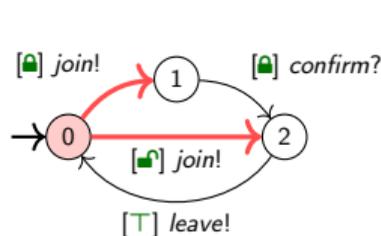
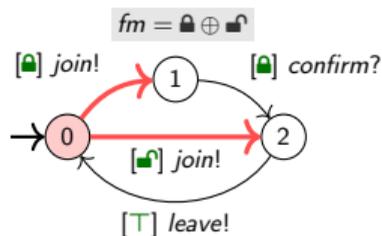
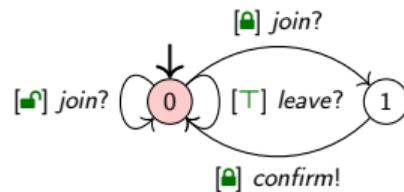
At state $(0, 0, 0)$:

$$[\text{lock} \vee \text{lock}] \text{rcp}(\{u_1\}, \text{join}) \wedge [] \text{rcp}(\{u_2\}, \text{join}) \wedge [] \text{rcp}(\{u_1, u_2\}, \text{join})$$

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

 $u_1 : \text{User}$  $u_2 : \text{User}$  $s : \text{Server}$

$$\text{fst}(\{\text{lock}\}, \text{join}) = ([1, 1], [1, 1]) \quad \text{fst}(\{\text{lock} \oplus \text{lock}\}, \text{join}) = ([1, *], [1, 1])$$

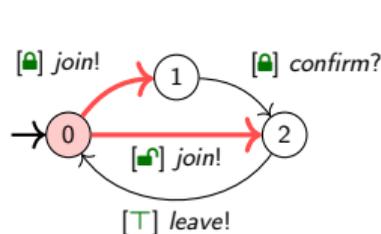
At state $(0, 0, 0)$:

$$[\text{lock} \vee \text{lock} \wedge fm] \text{rcp}(\{u_1\}, \text{join}) \wedge [] \text{rcp}(\{u_2\}, \text{join}) \wedge [] \text{rcp}(\{u_1, u_2\}, \text{join})$$

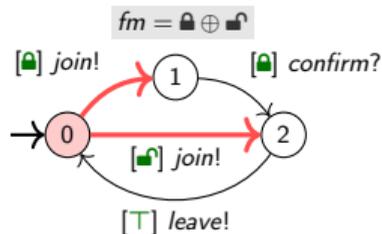
Featured receptiveness requirements are **constrained** with feature expression by:

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

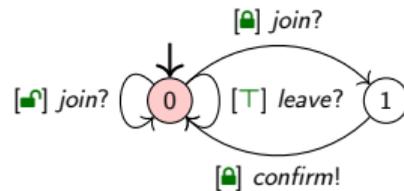
Featured Receptiveness Requirements



$u_1 : \text{User}$



$u_2 : \text{User}$



$s : \text{Server}$

$$\text{fst}(\{\text{lock}\}, \text{join}) = ([1, 1], [1, 1]) \quad \text{fst}(\{\text{server}\}, \text{join}) = ([1, *], [1, 1])$$

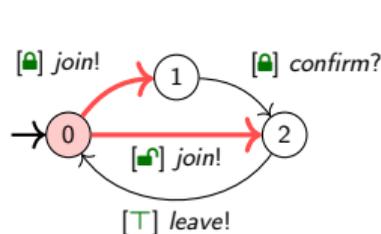
At state $(0, 0, 0)$:

$$[\text{lock} \vee \text{server} \wedge fm] \text{rcp}(\{u_1\}, \text{join}) \wedge [] \text{rcp}(\{u_2\}, \text{join}) \wedge [] \text{rcp}(\{u_1, u_2\}, \text{join})$$

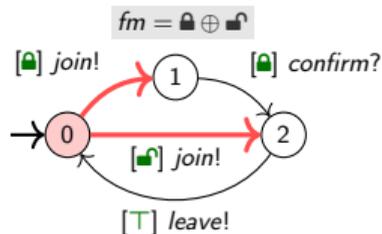
Featured receptiveness requirements are **constrained** with feature expression by:

- **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

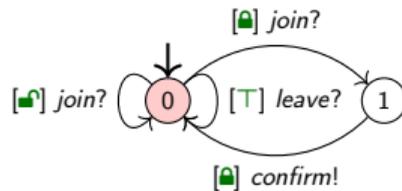
Featured Receptiveness Requirements



$u_1 : \text{User}$



$u_2 : \text{User}$



$s : \text{Server}$

$$\text{fst}(\{\text{lock}\}, \text{join}) = ([1, 1], [1, 1]) \quad \text{fst}(\{\text{user}\}, \text{join}) = ([1, *], [1, 1])$$

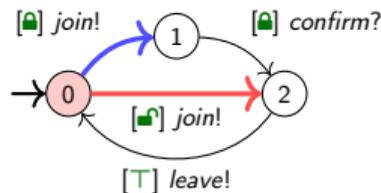
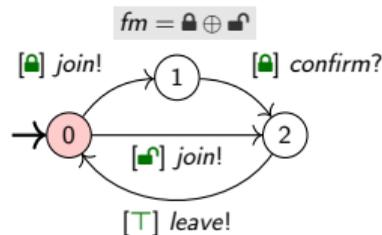
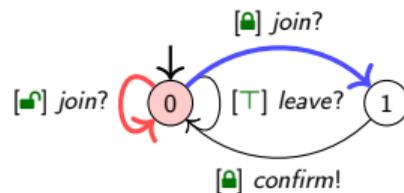
At state $(0, 0, 0)$:

$$[\text{lock} \vee \text{user} \wedge \text{fm}] \text{rcp}(\{u_1\}, \text{join}) \wedge [\text{fm}] \text{rcp}(\{u_2\}, \text{join}) \wedge [\text{lock} \vee \text{user} \wedge \text{user} \wedge \neg \text{lock} \wedge \text{fm}] \text{rcp}(\{u_1, u_2\}, \text{join})$$

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

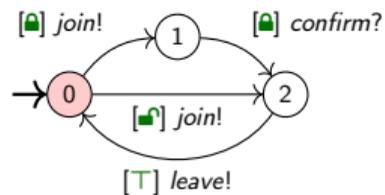
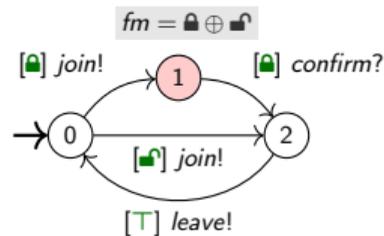
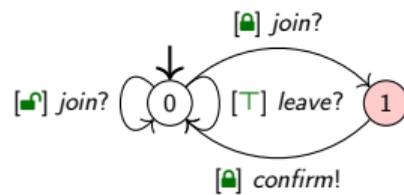
 $u_1 : \text{User}$  $u_2 : \text{User}$  $s : \text{Server}$ At state $(0, 0, 0)$:

$$\underline{[fm] \text{rcp}(\{u_1\}, \text{join})} \wedge [fm] \text{rcp}(\{u_2\}, \text{join}) \wedge [\text{lock} \wedge \neg \text{lock}] \text{rcp}(\{u_1, u_2\}, \text{join})$$

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

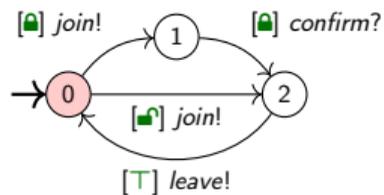
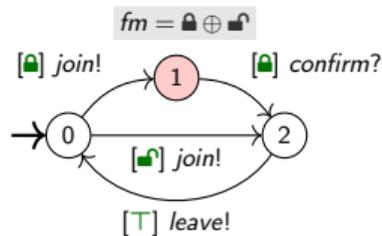
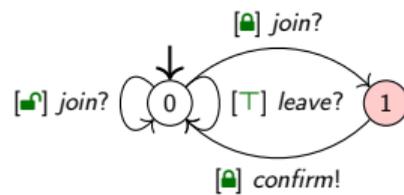
$$\{\text{lock}\} : (0, 0, 0) \xrightarrow{[\text{lock} \wedge fm] (\{u_1\}, \text{join}, \{s\})} \text{fst}[S](2, 0, 0)$$

Compliance with requirements

 $u_1 : \text{User}$  $u_2 : \text{User}$  $s : \text{Server}$ At state $(0, 1, 1)$:

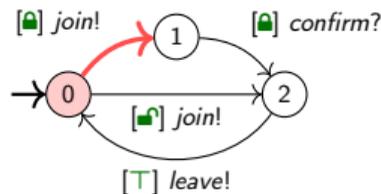
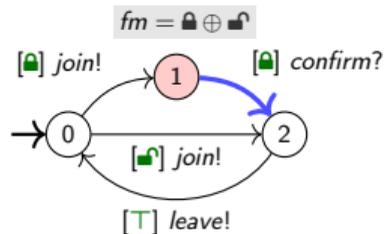
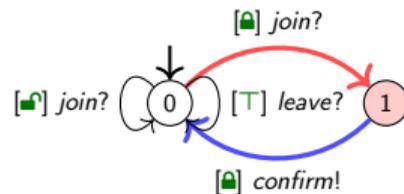
$$[\text{lock} \wedge \neg \text{unlock}] \text{rcp}(\{u_1\}, \text{join}) \wedge \dots$$

Compliance with requirements

 $u_1 : \text{User}$  $u_2 : \text{User}$  $s : \text{Server}$ At state $(0, 1, 1)$:

$$\neg [\text{lock} \wedge \neg \text{unlock}] \text{rcp}(\{u_1\}, \text{join}) \wedge \dots$$

Weak compliance with requirements

 $u_1 : \text{User}$  $u_2 : \text{User}$  $s : \text{Server}$ At state $(0, 1, 1)$:

$$[\text{lock} \wedge \neg \text{lock}] \text{rcp}(\{u_1\}, \text{join}) \wedge \dots$$

$$\{\text{lock}\} : (0, 1, 1) \xrightarrow{[\text{lock} \wedge \text{fm}](\{s\}, \text{confirm}, \{u_2\})} \text{fst}[S](0, 2, 0) \xrightarrow{[\text{lock} \wedge \text{fm}](\{u_1\}, \text{join}, \{s\})} \text{fst}[S](1, 2, 1)$$

Online prototype

- Specify
- Generate*
- Visualise
- Statistics

*SAT solver to solve *fm*

Online Tools for Featured Extensions

arcatools.org/assets/feta.html#onlinefeta

FETA Development Back to Area

FETA Specification

```

1 FCA user (confirm)(join,leave) = {
2   start 0
3   0 → 1 by join if s
4   1 → 2 by confirm if s
5   0 → 2 by join if o
6   2 → 0 by leave
7 }
8
9 FCA server (join,leave)(confirm) = {
10  start 0
11  0 → 1 by join if s
12  1 → 0 by confirm if s
13  0 → 0 by join if o
14  0 → 0 by leave
15 }
16
17 FS = (u1 → user, u2 → user, s → server)
18
19 FH = s xor o
20
21 FST = {
22   default = one to one // or 1..1 to 1..1
23   (o):join,leave = many to one // or 1..* to 1..1
24 }

```

FETA

FCA

user

server

FETA Examples

Auth Chat

FETA Information

Products: 2

- {s}, {o}

Featured System:

- # transitions: 142
- # states: 18

FETA:

- # transitions: 18
- # states: 8

FSTs:

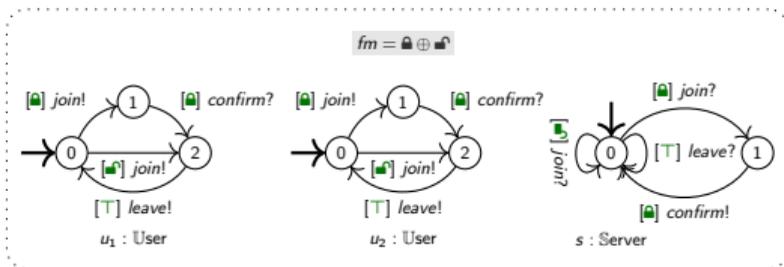
- fst(o){confirm} = 1->1
- fst(s){confirm} = 1->1
- fst(s){join} = 1->1
- fst(o){join} = 1..*->1
- fst(o){leave} = 1->1
- fst(o){leave} = 1..*->1

Copyright 2017-2021 - ARCA, U. Lameiro, pt

Wrapping up

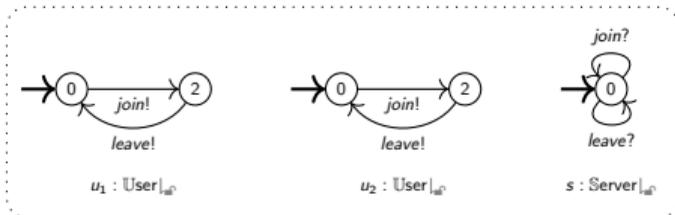
Featured Systems

FM'21:



Systems

ICTAC'20:



Featured Synchronisation Types

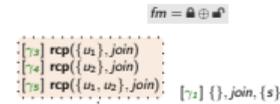
fst($\{a\}$, join) = ([1, *], [1, 1])
fst($\{a\}$, leave) = ([1, *], [1, 1])

st(join) = ([1, *], [1, 1])
st(leave) = ([1, *], [1, 1])

$\rightarrow = \rightarrow$
(diagram commutes)

Synchronisation Types

Featured Teams



[72] {u1}.join, {s}

{u1}.join, {s}

Teams

Featured Receptiveness

featured (weakly) receptive

(weakly) receptive

Main Theorem

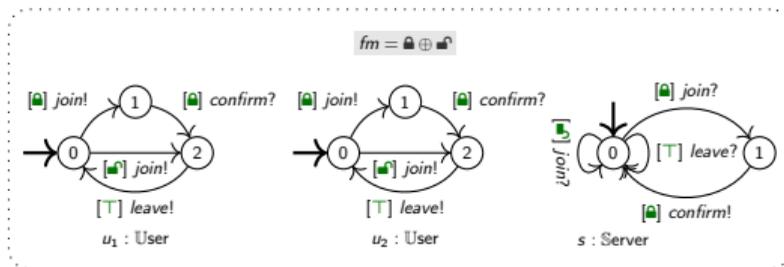
Receptiveness

Online prototype: <http://arcacools.org/feta>

Future and ongoing work

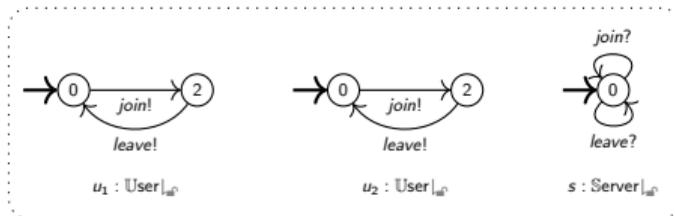
Featured Systems

FM'21:



Systems

ICTAC'20:



Featured Synchronisation Types

Compositionality

$$\text{fst}(\{a\}, \text{join}) = ([1, *], [1, 1])$$

$$\text{fst}(\{a\}, \text{leave}) = ([1, *], [1, 1])$$

$$\text{st}(\text{join}) = ([1, *], [1, 1])$$

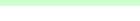
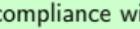
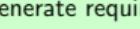
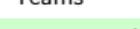
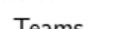
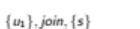
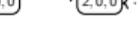
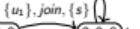
$$\text{st}(\text{leave}) = ([1, *], [1, 1])$$

$\rightarrow = \rightarrow$
(diagram commutes)

Synchronisation Types

Featured Teams

$$fm = a \oplus b$$



Featured Receptiveness

Smarter: which configurations derived compliant teams?

Featured Responsiveness

Main Theorem

(weakly) receptive

Receptiveness

Online prototype: <http://arcatools.org/feta>

Extensions (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