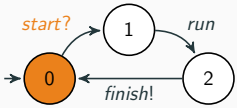
Can we Communicate?
Using Dynamic Logic to Verify Team Automata

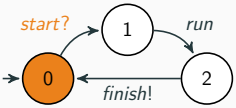
José Proença (CISTER – ISEP, Porto, Portugal)
& Maurice ter Beek (ISTI-CNR, Pisa, Italy)
& Guillermina Cledou (HASLab, INESC TEC, UM, Braga, Portugal)
& Rolf Hennicker (Ludwig-Maximilians-Universität, München, Germany)

March 7
Formal Methods 2023

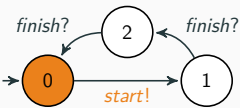
This talk



$\mathcal{R}unner_1$



$\mathcal{R}unner_2$



Controller

Multiple synchronisation

Team Automata

[FM'03,21] [CSCW'03] [ICTAC'20]

[COORDINATION'17,'20]

Properties

Interactions do not get stuck!

Automated verification

Properties as dynamic logic

Model-Checking with mCRL2

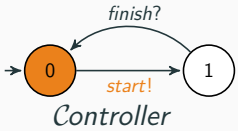
`http://arcatools.org/feta`

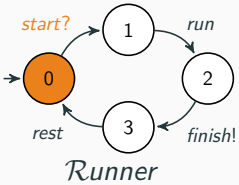
mCRL2 analysing system behaviour

We need to talk!

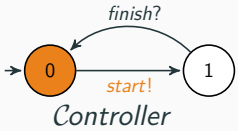
Verifying if interactions do not get stuck

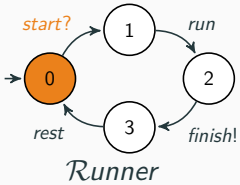
Interacting systems: Simple Race





Interacting systems: Simple Race

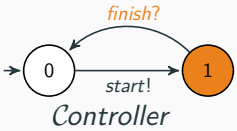


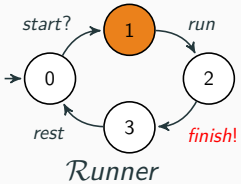


Receptiveness

- *If I can talk, will someone listen?*
- Will I be able to ask initially to start the Race?
- At any point:
  if I try to send some message, will it be received?

Interacting systems: Simple Race





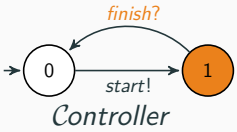
Receptiveness

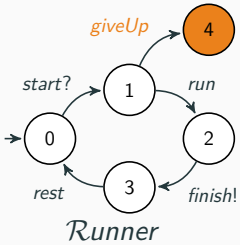
- *If I can talk, will someone listen?*
- Will I be able to ask initially to start the Race?
- At any point:
  if I try to send some message, will it be received?

(Weak) Responsiveness

- *If I am listening, will I succeed?*
- After asking to start, will I hear if you have finished?
- At any point:
  if I'm ready to receive messages, will any be sent?

Interacting systems: Simple Race





Receptiveness

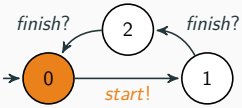
- *If I can talk, will someone listen?*
- Will I be able to ask initially to start the Race?
- At any point:
  if I try to send some message, will it be received?

Responsiveness

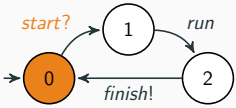
- *If I am listening, will I succeed?*
- After asking to start, will I hear if you have finished?
- At any point:
  if I'm ready to receive messages, will any be sent?

Team automata: multiple synchronisation

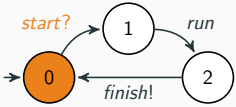
Multiple synchronisation: Race with 2 runners



Controller

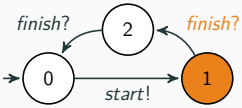


$\mathcal{R}unner_1$

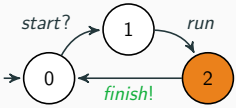


$\mathcal{R}unner_2$

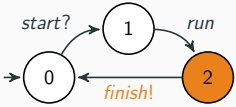
Multiple synchronisation: Race with 2 runners



Controller

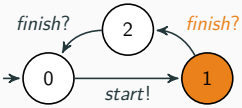


$\mathcal{R}unner_1$

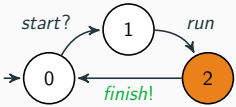


$\mathcal{R}unner_2$

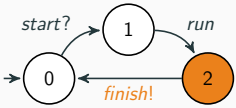
Multiple synchronisation: Race with 2 runners



$\mathcal{C}ontroller$



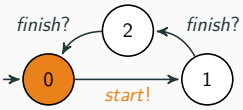
$\mathcal{R}unner_1$



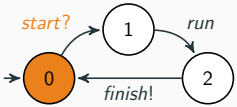
$\mathcal{R}unner_2$

start: $1 \to 2$ finish: $1 \to 1$ run: internal

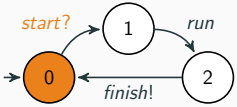
Multiple synchronisation: Race with 2 runners



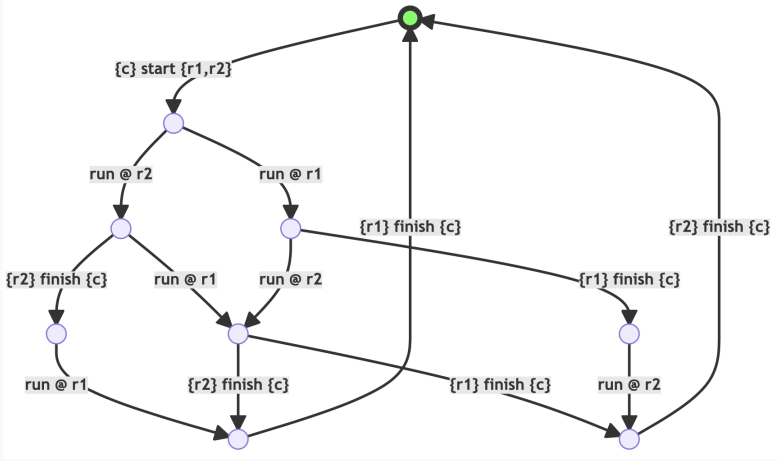
Controller



$\mathcal{R}unner_1$

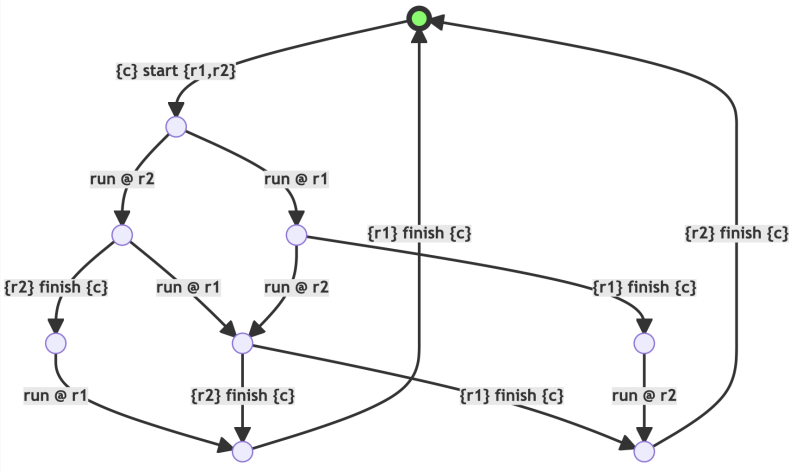


$\mathcal{R}unner_2$



start: $1 \to 2$ finish: $1 \to 1$ run: internal

Challenge for Dynamic Logic



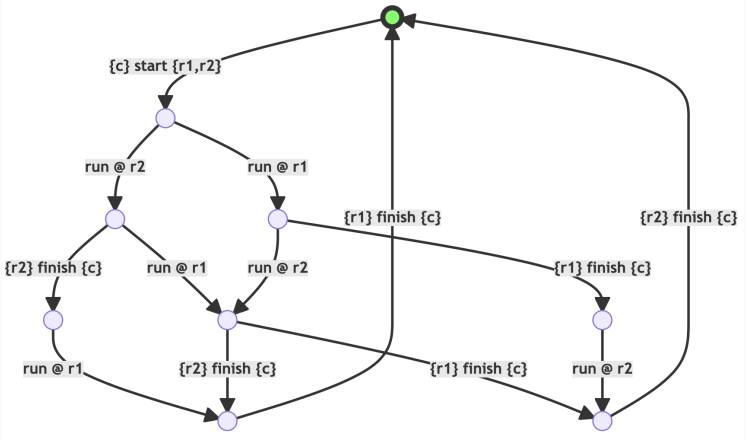
start: $1 \to 2$ finish: $1 \to 1$ run: internal

Formula that characterises receptiveness?

Φ such that:

$$\mathfrak{M} \models \Phi$$
$$\Leftrightarrow$$
$\mathfrak{M}$ **is receptive**

Need more information

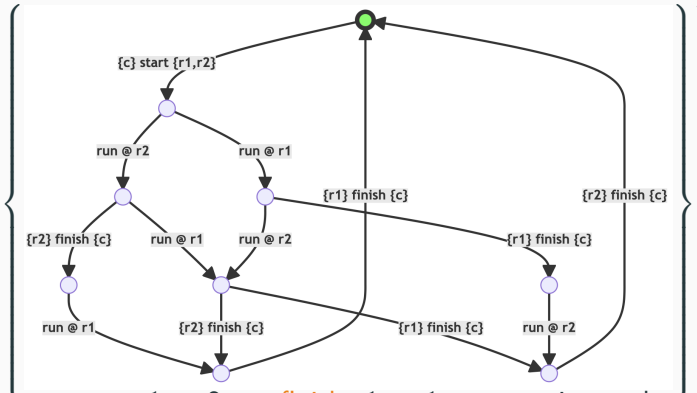


start: $1 \to 2$ finish: $1 \to 1$ run: internal

At each state:

- We only know which interactions can occur
- We do NOT know if anyone wanted to send (and failed)

Our approach



++

~~start~~: $1 \to 2$ ~~finish~~: $1 \to 1$ ~~run~~: ~~internal~~

start: ANY finish: ANY run: ANY

- Instead of:
  $\mathfrak{M} \models \Phi$
  $\begin{pmatrix} 9\ \textit{states} \\ 13\ \textit{trans} \end{pmatrix}$
- Find:
  $\mathfrak{M}^{++} \models$
  $[\textit{Allowed}^*]\,\Phi$
  $\begin{pmatrix} 27\ \textit{states} \\ 108\ \textit{trans} \end{pmatrix}$

Verifying receptiveness

Overview

$$\mathfrak{M}^{++} \models [Allowed^*]\Phi$$

Formal Definition

What is receptiveness?

Dynamic logic

Foundational detour

Receptiveness formulas

How to find Φ?

Verification with mCRL2

Receptiveness Definition

GIVEN

- Team
  $\langle A_1, A_2, \ldots \rangle$ + sync
- Reachable state
  $\langle q_1, q_2, \ldots \rangle$
- Action
  $a : outs \to ins$

IF

- $\exists \{A_i \mid i \in Snd\}$
  $\subseteq \{A_1, A_2, \ldots\}$
- $\{q_i \mid i \in Snd\}$
  can do $a!$
- $|Snd|$
  $\in outs$

THEN

- $\exists \{A_j \mid j \in Rcv\}$
  $\subseteq \{A_1, A_2, \ldots\}$
- $\{q_j \mid j \in Rcv\}$
  can do $a?$
- $|Rcv|$
  $\in ins$

Receptiveness Definition

GIVEN

- Team $\langle A_1, A_2, \ldots \rangle$ + sync
- Reachable state $\langle q_1, q_2, \ldots \rangle$
- Action $a : outs \to ins$

IF

- $\exists \{A_i \mid i \in Snd\} \subseteq \{A_1, A_2, \ldots\}$
- $\{q_i \mid i \in Snd\}$ can do $a!$
- $|Snd| \in outs$

THEN

- $\exists \{A_j \mid j \in Rcv\} \subseteq \{A_1, A_2, \ldots\}$
- $\{q_j \mid j \in Rcv\}$ can do $a?$
- $|Rcv| \in ins$

In our Race example

Controller can start!	$\Rightarrow$	both *Runners* must be able to start?
$Runner_1$ can finish!	$\Rightarrow$	*Controller* can finish?
$Runner_2$ can finish!	$\Rightarrow$	*Controller* can finish?

Process Logic + regular expressions

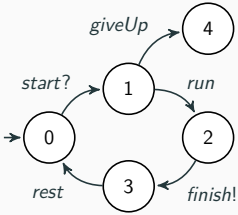
Dynamic Logic (without propositions and tests)

$$\phi \ ::= \ \text{true} \mid \text{false} \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid \phi_1 \to \phi_2 \mid \langle\alpha\rangle\,\phi \mid [\alpha]\,\phi$$

where $\alpha \in ACT$ are structured actions over a set Act:

$$\alpha := a \in Act \mid \alpha;\alpha \mid \alpha + \alpha \mid \alpha^*$$

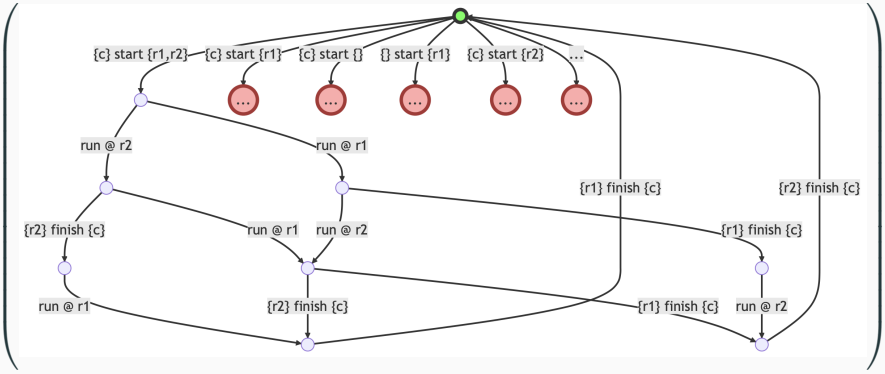
Properties of our *faulty* runner



- $\langle start? \rangle$ true
- $\langle -^*; run \rangle$ true
- $[-^*; start?; finish!]$ false
- $[-^*; start?] \langle -^*; (finish! + giveUp) \rangle$ true

Verifying receptiveness

$\mathfrak{M}^{++}$



$$[\mathit{Allowed}^*]$$

$$\models \left(\begin{array}{rcl} \langle \{c\}\ \mathit{start}\ \{\} \rangle \mathsf{true} & \Rightarrow & \langle \{c\}\ \mathit{start}\ \{r1, r2\} \rangle \mathsf{true} \wedge \\ \langle \{r1\}\ \mathit{finish}\ \{\} \rangle \mathsf{true} & \Rightarrow & \langle \{r1\}\ \mathit{finish}\ \{c\} \rangle \mathsf{true} \wedge \\ \langle \{r2\}\ \mathit{finish}\ \{\} \rangle \mathsf{true} & \Rightarrow & \langle \{r2\}\ \mathit{finish}\ \{c\} \rangle \mathsf{true} \end{array} \right)$$

Verifying receptiveness

$$\mathfrak{M}^{++} \models [Allowed^*] \left(\begin{array}{lcl} \langle \{c\}\ start\ \{\}\rangle \text{true} & \Rightarrow & \langle \{c\}\ start\ \{r1, r2\}\rangle \text{true} \wedge \\ \langle \{r1\}\ finish\ \{\}\rangle \text{true} & \Rightarrow & \langle \{r1\}\ finish\ \{c\}\rangle \text{true} \wedge \\ \langle \{r2\}\ finish\ \{\}\rangle \text{true} & \Rightarrow & \langle \{r2\}\ finish\ \{c\}\rangle \text{true} \end{array} \right)$$

Responsiveness

$$\mathfrak{M}^{++} \models [Allowed^*] \left(\begin{array}{l} (\langle \{\}\ finish\ \{c\}\ +\ \{\}\ start\ \{r1, r2\}\rangle \text{true}) \\ \quad \Rightarrow\ (\langle \{c\}\ start\ \{r1, r2\}\ +\ \{r1\}\ finish\ \{c\}\ +\ \{r2\}\ finish\ \{c\}\rangle \text{true}) \end{array} \right)$$

Verifying receptiveness

$$\mathfrak{M}^{++} \models [Allowed^*] \left(\begin{array}{rcl} \langle \{c\}\ start\ \{\}\rangle \text{true} & \Rightarrow & \langle \{c\}\ start\ \{r1, r2\}\rangle \text{true} \wedge \\ \langle \{r1\}\ finish\ \{\}\rangle \text{true} & \Rightarrow & \langle \{r1\}\ finish\ \{c\}\rangle \text{true} \wedge \\ \langle \{r2\}\ finish\ \{\}\rangle \text{true} & \Rightarrow & \langle \{r2\}\ finish\ \{c\}\rangle \text{true} \end{array} \right)$$

Also...

Verified automatically (and efficiently) using the mCRL2 model checker

Proved that this notion of receptiveness (and others) via Dynamic logic matches the ones in the literature, *(e.g., compatibility notions in [Alfaro, Henzinger 2005] (interface automata) or [Carmona, Kleijn 2013] (multi-component environment))*

Implementation – http://arcatools.org/feta

(F)ETA Specification

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

Race example

(F)ETA Examples

Simple (ETA) | Race (ETA) | Chat (ETA) | Auth (FETA)

(F)ETA diagram

(F)System diagram

(F)CA

View mCRL2 evidence

Verification in mCRL2

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:

mCRL2 analysing system behaviour

Implementation –

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));
```

(F)ETA Specification

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

Race example

(F)ETA Examples

Simple (ETA) | Race (ETA) | Chat (E

```
finish|c_finish + r1_run)* ](
```

```
finish|c_finish + r1_run)* ](
```

mCRL2

alysing system behaviour

(F)ETA Specific

```
1  //Race exa
2  CA runner
3   start 0
4   0 --> 1 by
5   1 --> 2 by
6   2 --> 0 by
7  }
8
9  CA control
10  start 0
11  0 --> 1 by
12  1 --> 2 by
13  2 --> 0 by
14 }
15
16 S = (r1:ru
17
18 STS = {
19  default =
20  start = 1
21 }
```

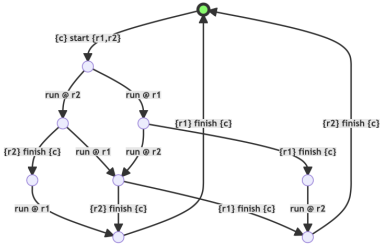
Race example

(F)ETA Exampl

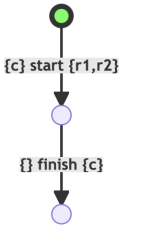
Simple (ETA) Ra

View mCRL2 evidence

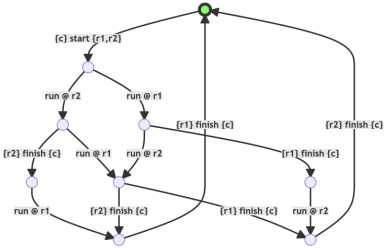
Receptiveness: true



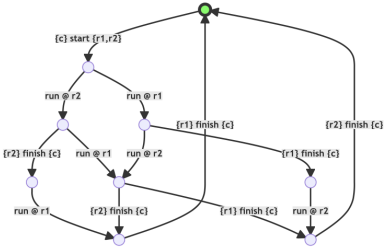
Responsiveness: false



Weak Receptiveness: true



Weak Responsiveness: true



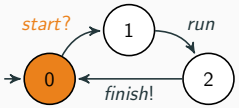
+ r1_run)*](

+ r1_run)*](

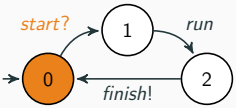
RL2

m behaviour

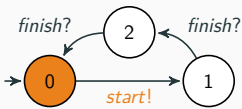
Wrap up



$\mathcal{R}unner_1$



$\mathcal{R}unner_2$



$\mathcal{C}ontroller$

Multiple synchronisation

Team Automata

[FM'03,21] [CSCW'03] [ICTAC'20] [COORDINATION'17,'20]

Properties

Interactions do not get stuck!

Receptiveness & Responsiveness

Automated verification

Properties as dynamic logic

Model-Checking with mCRL2

`http://arcatools.org/feta`

mCRL2 analysing system behaviour