Deciding Contextual Equivalence for IMJ*

Andrzej Murawski Steven Ramsay Nikos Tzevelekos

University of Warwick and Queen Mary University of London

$$\Delta | \Gamma \vdash t_1 \cong t_2 : \theta$$

For all interface tables $\Delta' \supseteq \Delta$ and IMJ contexts $C\square$ such that:

$$\Delta' | \varnothing \vdash C[t_i] : \text{void}$$

it follows that:

$C[t_1]$ *terminates* *iff* $C[t_2]$ *terminates*

```
new {_:I; run: λ_. div}
```

```
let x = new {_: IntRef;} in
new {_:I;
  run: λ_.
    if x.val = 0 then
      x.val := 1;
      f.run();
      if x.val = 2 then
        skip
      else
        div
    else if x.val = 1 then
      x.val := 2
    else
      div
}
```

```
new {_:I; run: λ_. div}
```

I = { run: void → void }

```
let
new {_:I;
  run: λ_.
    if x.val = 0 then
      x.val := 1;
      f.run();
      if x.val = 2 then
        skip
      else
        div
    else if x.val = 1 then
      x.val := 2
    else
      div
}
```

```
new {_:I; run: λ_. div}
```

```
let x = new {_: IntRef;} in
new {_:I;
  run: λ_.
    if x.val = 0 then
      x.val := 1;
      f.run();
      if x.val = 2 then
        skip
      else
        div
    else if x.val = 1 then
      x.val := 2
    else
      div
}
```

```
IntRef = { val: int }
```

```
new {_:I; run: λ_. div}
```

```
let x = new {_: IntRef;} in
new {_:I;
  run: λ_.
    if x.val = 0 then
      x.val := 1;
      f.run();
      if x.val = 2 then
        ()
      e
    else if x.val = 1 then
      x.val := 2
    else
      div
}
```

f : I

```
new {_:I; run: λ_. div}
```

```
let x = new {_: IntRef;} in
new {_:I;
  run: λ_.
    if x.val = 0 then
      x.val := 1;
      f.run();
      if x.val = 2 then
        skip
      else
        div
    else if x.val = 1 then
      x.val := 2
    else
      div
}
```

```
let z = new { _:IRef;} in
let f =
 new {_:I; run: λ_. z.val.run() }
in
z.val := □
z.val.run()
```

```
new {_:I; run: λ_. div}
```

```
let x = new {_: IntRef;} in
new {_:I;
  run: λ_.
    if x.val = 0 then
      x.val := 1;
      f.run();
      if x.val = 2 then
        skip
      else
        div
    else if x.val = 1 then
      x.val := 2
    else
      div
}
```

```
let z = new { _:IRef;} in
let f =
                                  run() }
z.val.run()
```

```
IRef = { val: I }
```

```
new {_:I; run: λ_. div}
```

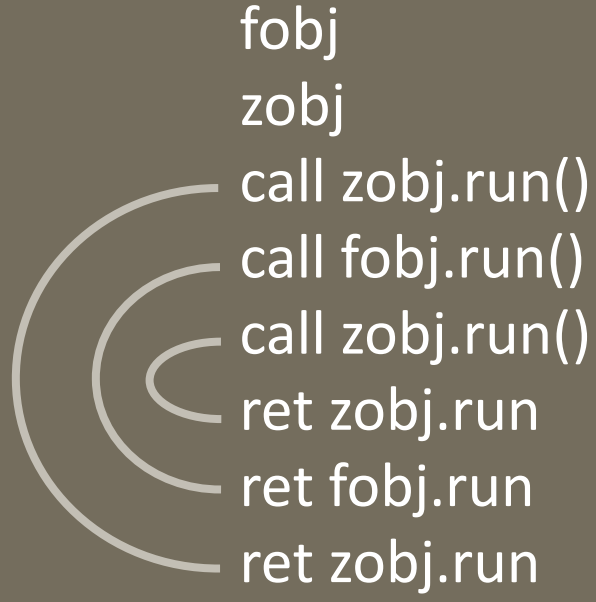
```
let x = new {_: IntRef;} in
new {_:I;
  run: λ_.
    if x.val = 0 then
      x.val := 1;
      f.run();
      if x.val = 2 then
        skip
      else
        div
    else if x.val = 1 then
      x.val := 2
    else
      div
}
```

```
let z = new { _:IRef;} in
let f =
 new {_:I; run: λ_. z.val.run() }
in
z.val := □
z.val.run()
```

```
new {_:I; run: λ_. div}
```

```
let x = new {_: IntRef;} in
new {_:I;
  run: λ_.
    if x.val = 0 then
      x.val := 1;
      f.run();
      if x.val = 2 then
        skip
      else
        div
    else if x.val = 1 then
      x.val := 2
    else
      div
}
```

```
let z = new { _:IRef;} in
let f =
 new {_:I; run: λ_. z.val.run() }
in
z.val := □
z.val.run()
```



```
let x = new {_:IntRef;} in
let c1 = new {_:ObjRef;} in
let c2 = new {_:ObjRef;} in
new {_:ObjCell;
  get: λ_.if x.val then c1.val else c2.val,
  getprev: λ_.if x.val then c2.val else c1.val,
  set: λo.
    if x.val then x.val := 0 else x.val := 1;
    if x.val then c1.val := o else c2.val := o
}
```

```
let last = new {_:ObjRef;} in
let current = new {_:ObjRef;} in
new {_:ObjCell;
  get: λ_.current.val,
  getprev: λ_.last.val,
  set: λo.last.val := current.val; current.val := o
}
```

IMJ*

[MRT ATVA'15]

null n skip x $exp = exp$ $exp + exp$ $(I)exp$

$exp.fld$ $exp.fld := exp$ $exp.m(exp_1,\ldots,exp_k)$ $exp\ ;\ exp$

let x = exp **in** exp **if** x = exp **then** exp **else** exp **while** exp **do** exp

new { *this*: I; m_1: $body_1$, …, m_k: $body_k$ }

IMJ*

[MRT ATVA'15]

Only finite types, ground fields

Only second-order objects

$$\Delta \mid \Gamma \vdash t : \theta$$

Only first-order objects

Only iteration

CONEQCT [MRT ATVA'15 (TOOL)]

1 Translate IMJ* terms into their strategies in the game model, represented as two IMJ Automata (IMJA).

[MT POPL'14]

2 Reduce the equivalence problem for IMJA to the emptiness problem for Fresh Pushdown Register Automata (FPDRA).

[MRT ATVA'15]

3 Solve the emptiness problem for FPDRA using saturation algorithm.

[MRT MFCS'14]

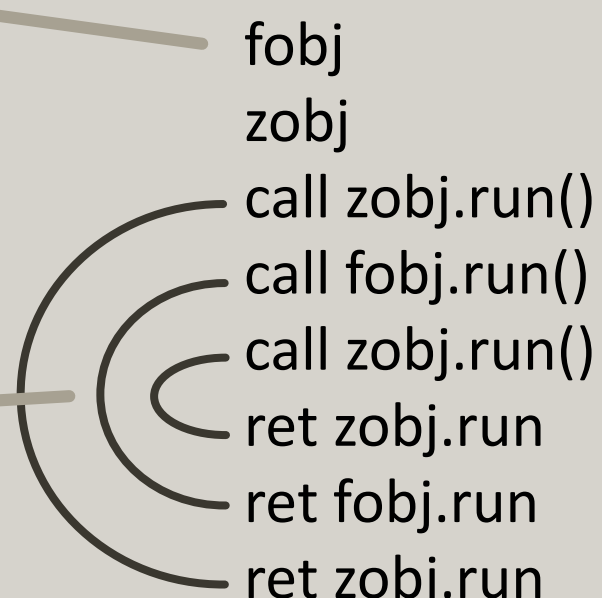
IMJA: A machine representation for strategies (sets of plays).

Object creation

Fresh-name recognition

Call stack discipline

Visible pushdown stack



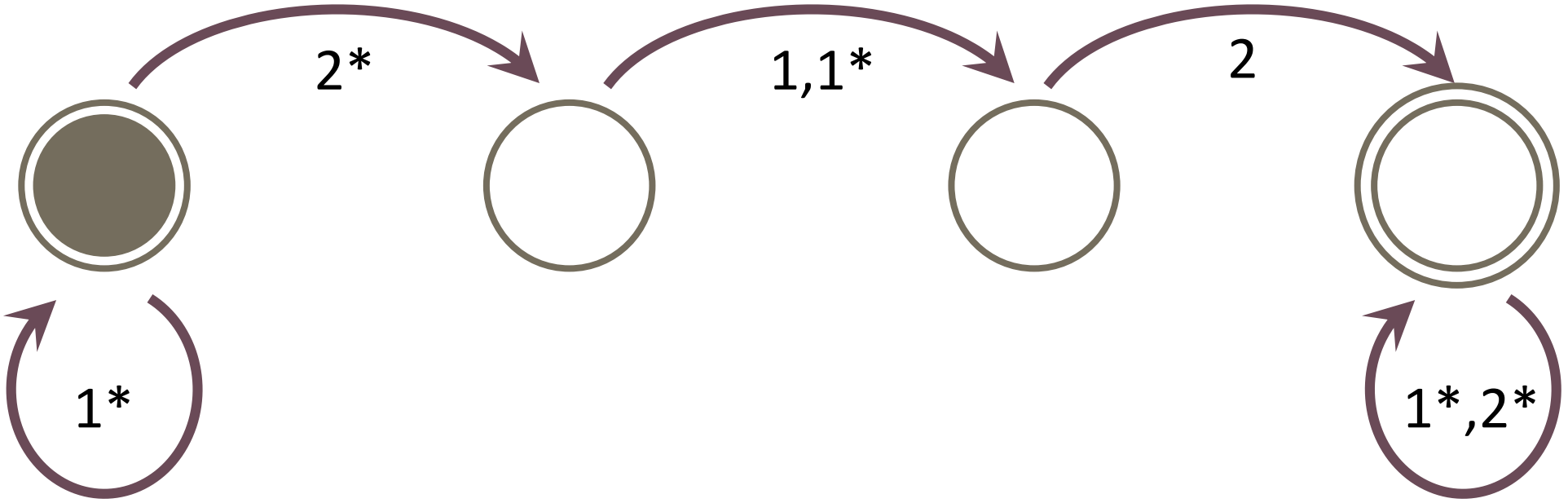
Finite set of possible moves modulo object names

Accepts words over a *nominal* alphabet

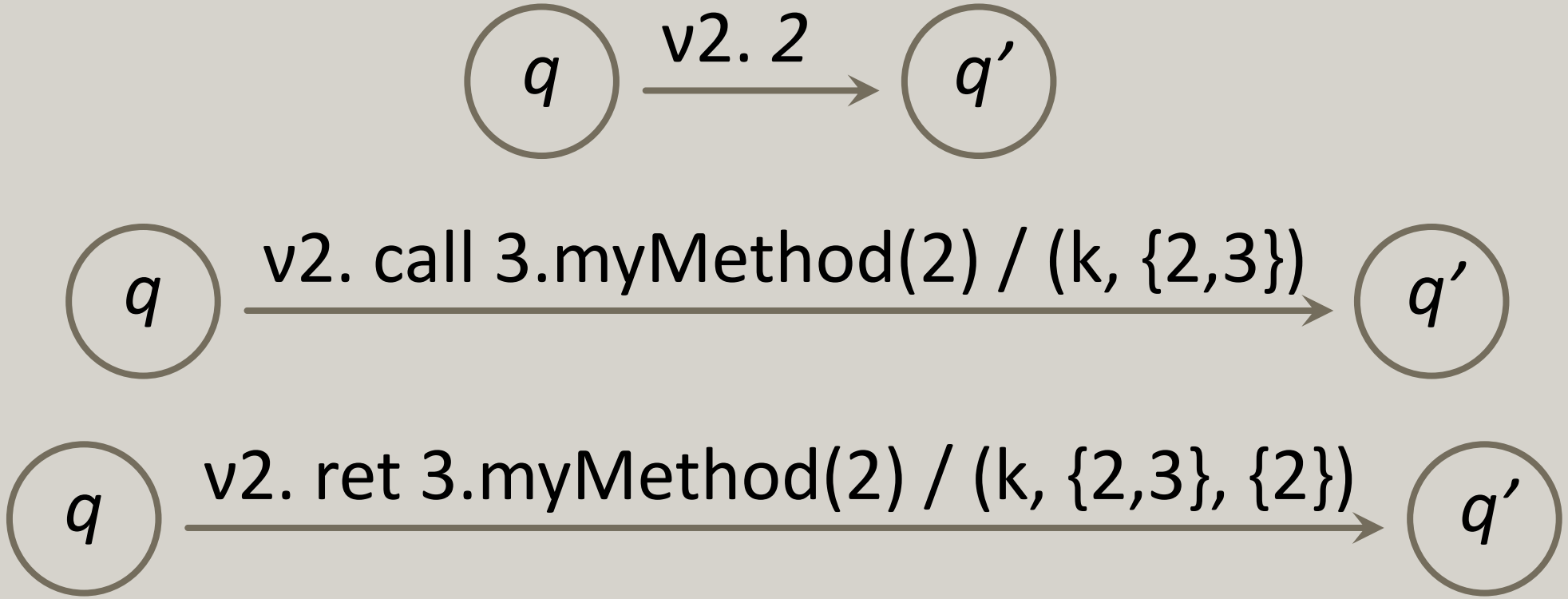
(Representation of stores not shown)

1-Slide intro to RA

$$\mathcal{L}(\mathcal{A}) = \{w \in \Sigma^* \mid \forall i.\, \sigma(i) \neq \sigma(i+1) \wedge \exists i, j.\, i \neq j \wedge \sigma(i) = \sigma(j)\}$$

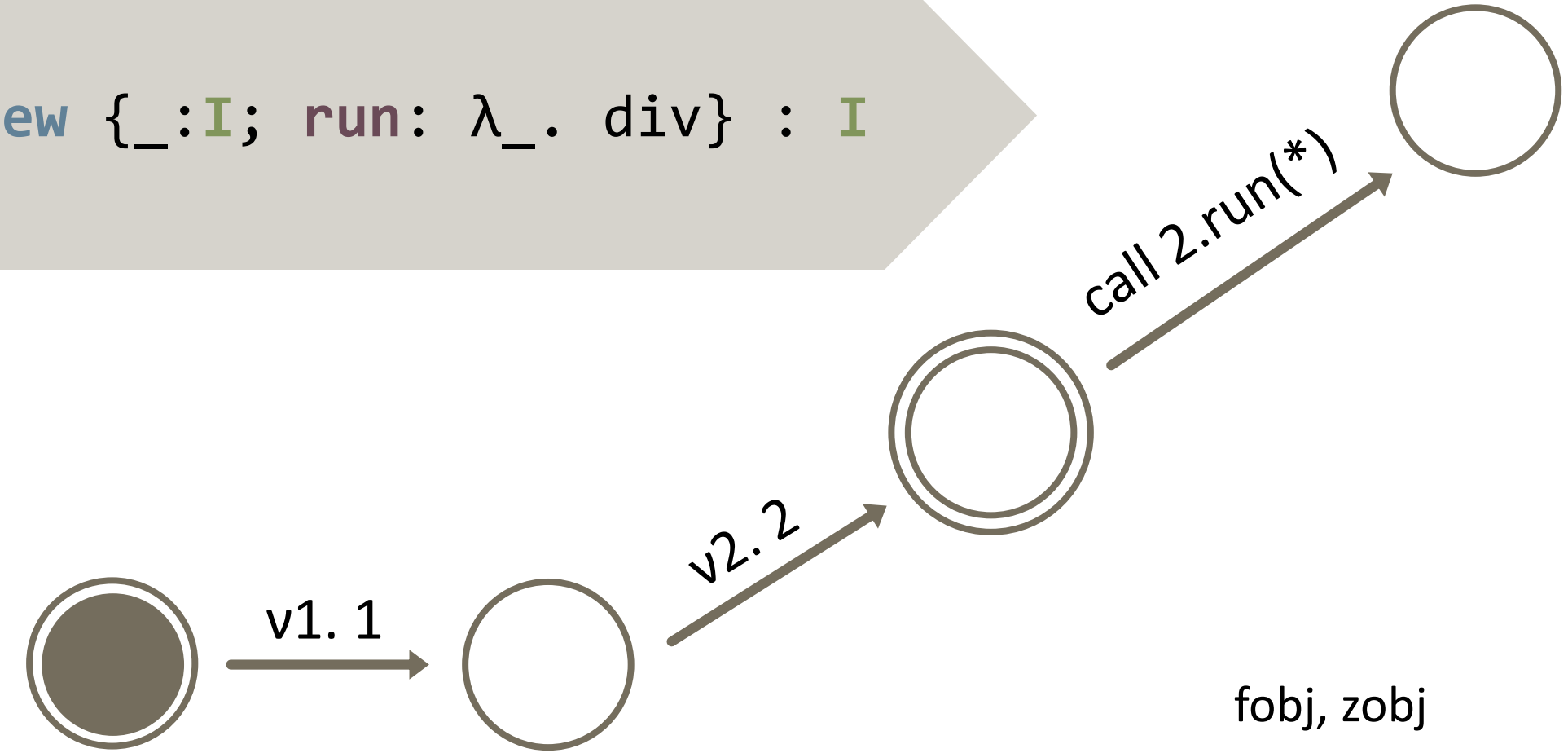


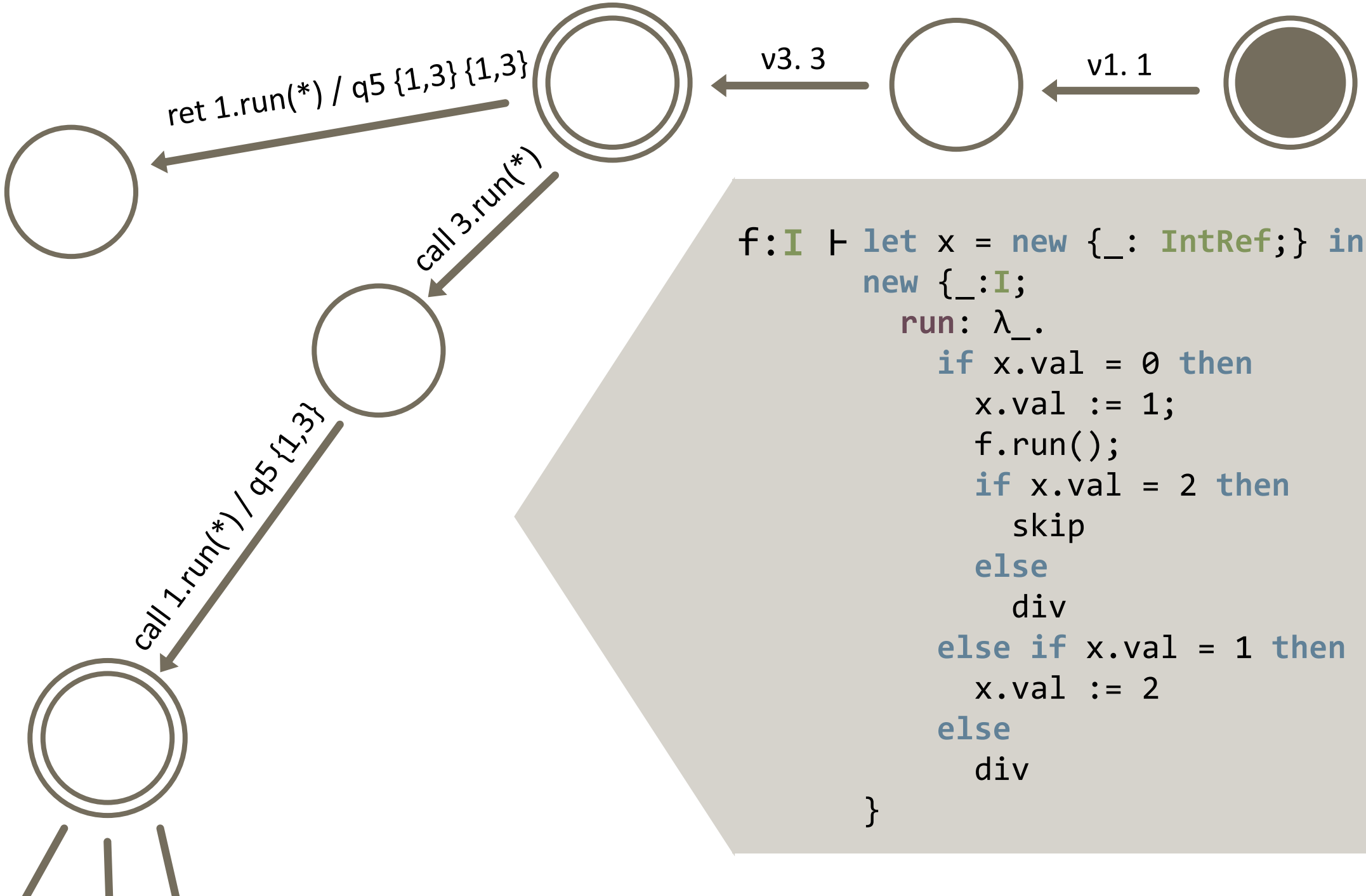
IMJA: A machine representation for strategies (sets of plays).



+ Bookkeeping

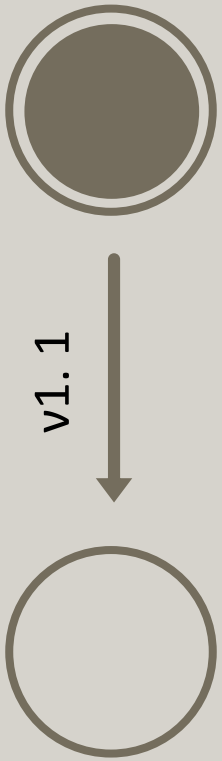
```
f:I ⊢ new {_:I; run: λ_. div} : I
```

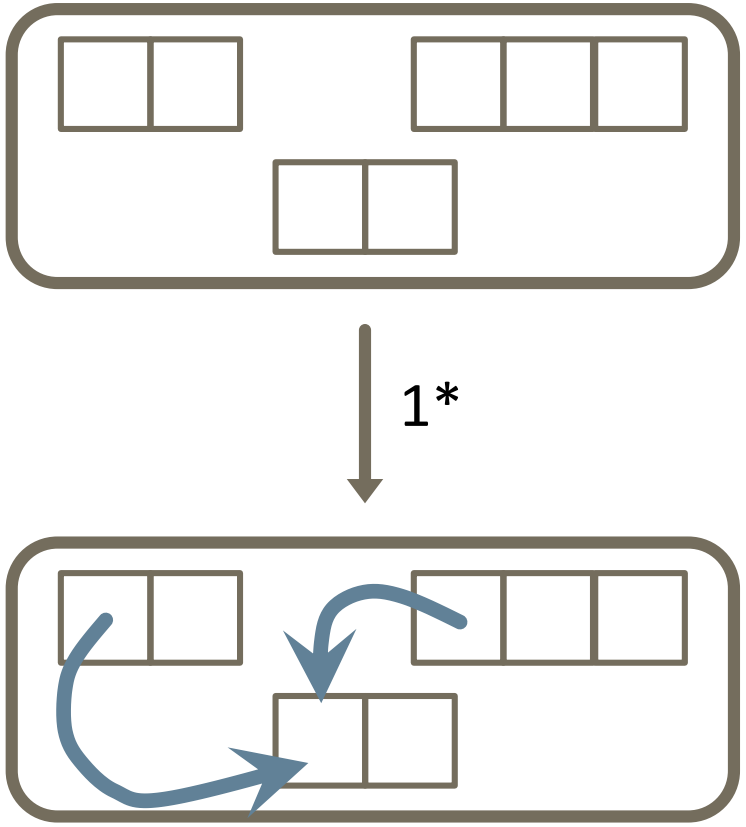


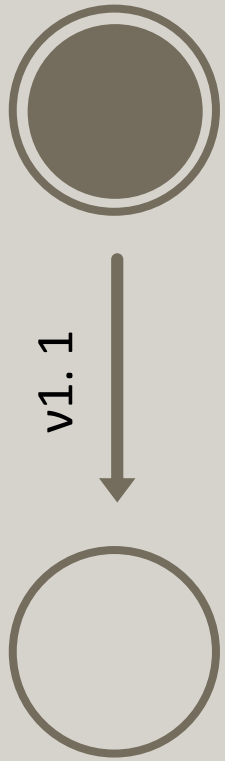



```
f:I ⊢ let x = new {_: IntRef;} in
      new {_:I;
        run: λ_.
          if x.val = 0 then
            x.val := 1;
            f.run();
            if x.val = 2 then
              skip
            else
              div
          else if x.val = 1 then
            x.val := 2
          else
            div
      }
```

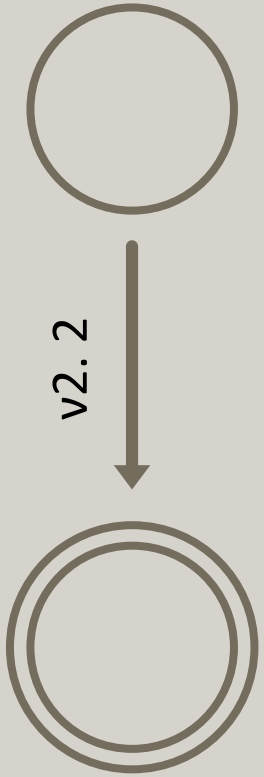
SYNCHRONISATION

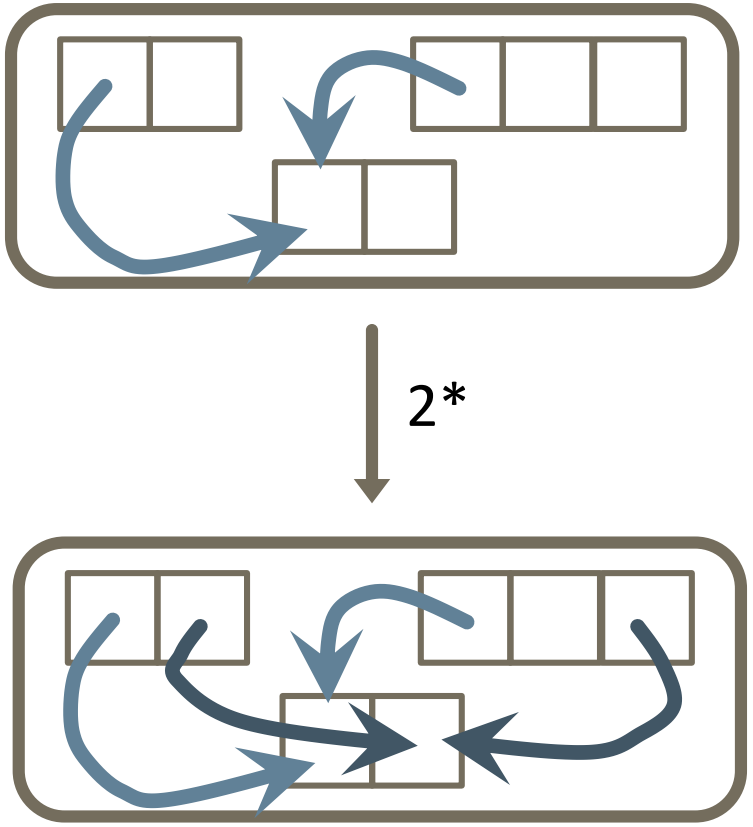


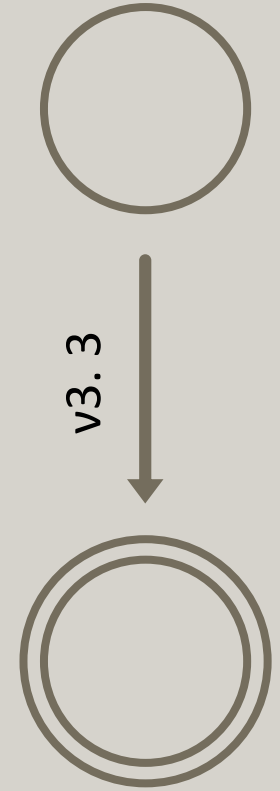




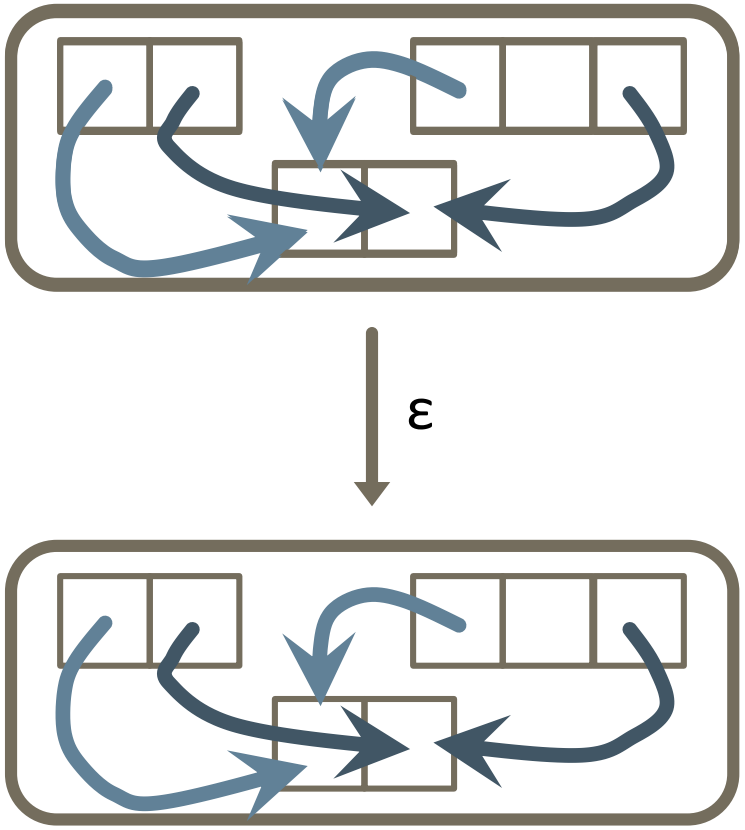
SYNCHRONISATION



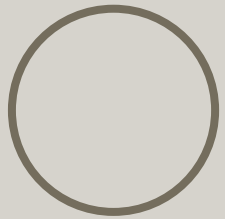


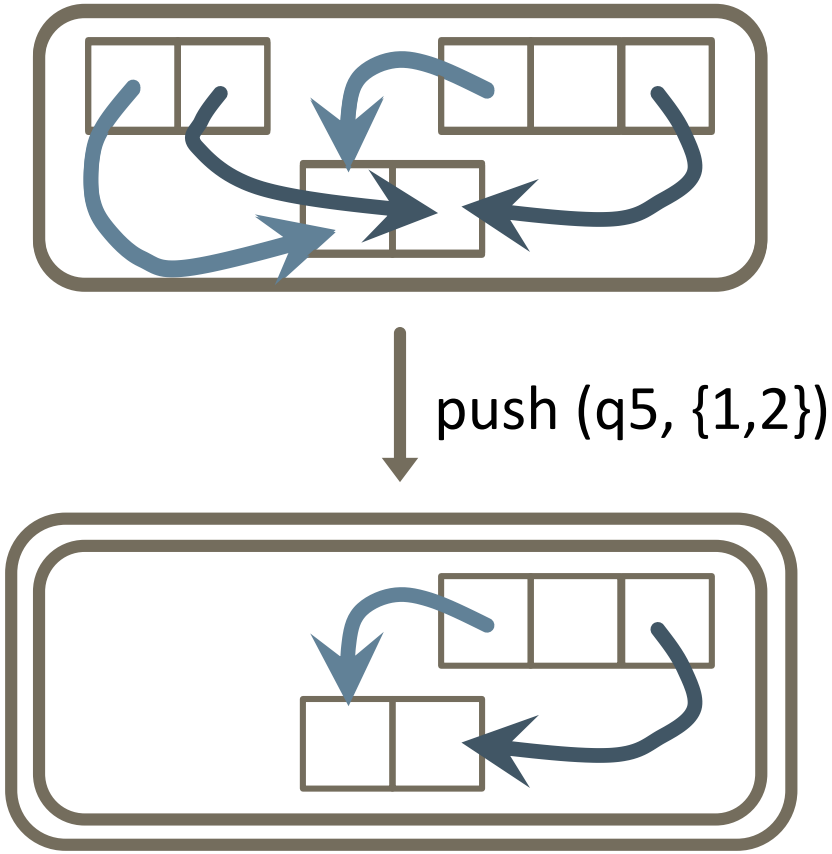


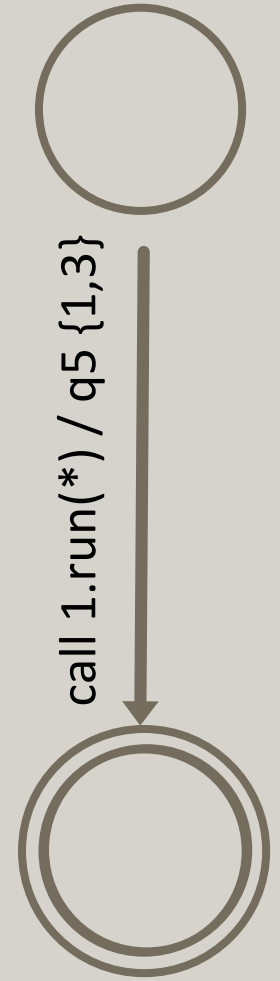
SYNCHRONISATION



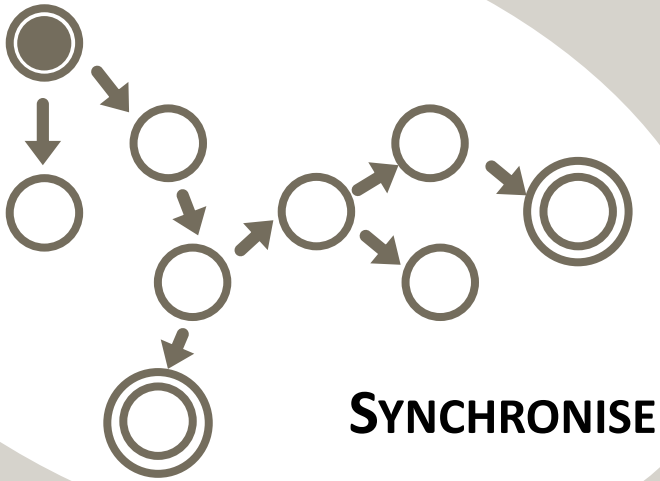
SYNCHRONISATION



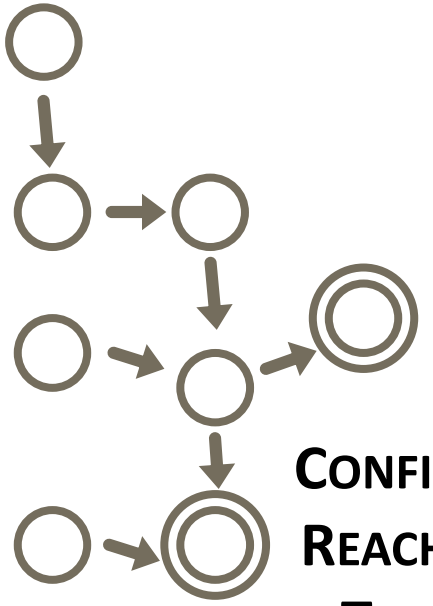




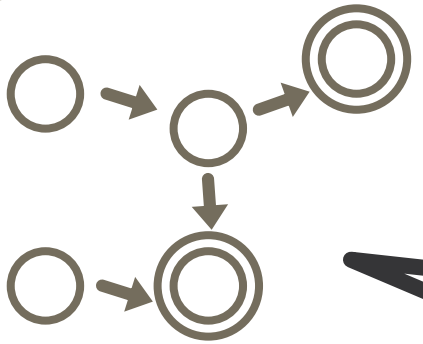
SATURATION



SYNCHRONISED SYSTEM (FPRDA)



CONFIGURATIONS REACHING SOME TARGET (RA)



TARGET CONFIGURATIONS (RA)

$aabbcd \in \mathcal{L}(\mathcal{A}, q)$ implies $(q, aabbcd)$ is a target

CONEQCT [MRT ATVA'15 (TOOL)]

1 Translate IMJ* terms into their strategies in the game model, represented as two IMJ Automata (IMJA).

[MT POPL'14]

2 Reduce the equivalence problem for IMJA to the emptiness problem for Fresh Pushdown Register Automata (FPDRA).

[MRT ATVA'15]

3 Solve the emptiness problem for FPDRA using saturation algorithm.

[MRT MFPS'15]

FUTURE WORK

Enlarging IMJ*

Tool enhancements

Tool optimisation