An encoding of Distributed PlusCal in the Join Calculus

By Ghilain BERGERON (Université de Lorraine, CNRS, Inria, Loria)

On August 12, 2024

Motivation

- Formally describing the behaviors of distributed algorithms is hard.

Motivation

- Formally describing the behaviors of distributed algorithms is hard.
- Correctly implementing such algorithms is way harder.

Motivation

- Formally describing the behaviors of distributed algorithms is hard.
- Correctly implementing such algorithms is way harder.
- One solution: deriving a working program through the use of a certified specification compiler.

Some context

∗ A C-like language for specifying (concurrent) systems

Some context

- A C-like language for specifying (concurrent) systems
- Compiles down to a TLA+ specification

Some context

- A C-like language for specifying (concurrent) systems
- Compiles down to a TLA+ specification
  - Allows the system to be checked…

Some context

- A C-like language for specifying (concurrent) systems
- Compiles down to a TLA+ specification
  - Allows the system to be checked…
  - …by model checkers (TLC, Apalache, …)

Some context

- A C-like language for specifying (concurrent) systems
- Compiles down to a TLA+ specification
  - Allows the system to be checked...
  - ...by model checkers (TLC, Apalache, ...)
  - ...by interactive proof assistants (TLAPS, ...)

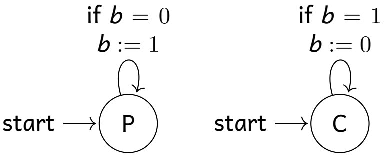
Some context

- A C-like language for specifying (concurrent) systems
- Compiles down to a TLA+ specification
  - Allows the system to be checked...
  - ...by model checkers (TLC, Apalache, ...)
  - ...by interactive proof assistants (TLAPS, ...)

```
(*--algorithm Alternate {
    variable b = 0;

    process (P = 0) {
t0:     while (TRUE) {
            await b = 0;
            b := 1;
        }
    }

    process (C = 1) {
t1:     while (TRUE) {
            await b = 1;
            b := 0;
        }
    }
}*)
```

Some context

- A C-like language for specifying (concurrent) systems
- Compiles down to a TLA+ specification
  - Allows the system to be checked...
  - ...by model checkers (TLC, Apalache, ...)
  - ...by interactive proof assistants (TLAPS, ...)



```
(*--algorithm Alternate {
  variable b = 0;

  process (P = 0) {
t0: while (TRUE) {
      await b = 0;
      b := 1;
    }
  }

  process (C = 1) {
t1: while (TRUE) {
      await b = 1;
      b := 0;
    }
  }
}*)
```

Some context

Extends PlusCal with:

Some context

Extends PlusCal with:

- communication primitives

Some context

Extends PlusCal with:

- communication primitives
  - Channels, FIFOs (ordered channels)

Some context

Extends PlusCal with:

- communication primitives
  - Channels, FIFOs (ordered channels)
  - Send, receive

Some context

Extends PlusCal with:

- communication primitives
  - Channels, FIFOs (ordered channels)
  - Send, receive
- and a distinction between local and distant concurrency

Some context

- A distributed process calculus

Some context

- A distributed process calculus
- Better viewed as a programming model rather than a calculus

Some context

- A distributed process calculus
- Better viewed as a programming model rather than a calculus
- Based on chemical phenomena

Some context

- Atoms $A ::= \alpha\langle e_1, \ldots, e_n\rangle$

Some context

- Atoms $A ::= \alpha\langle e_1, \ldots, e_n\rangle$
- Molecules $M ::= A_1 \mid \cdots \mid A_n$

Some context

- Atoms $A ::= \alpha\langle e_1, \ldots, e_n\rangle$
- Molecules $M ::= A_1 \mid \cdots \mid A_n$
- Joins $J ::= \alpha_1\langle x_{11}, \ldots, x_{1m_1}\rangle \mid \cdots \mid \alpha_n\langle x_{n1}, \ldots, x_{nm_n}\rangle$

Some context

- Atoms $A ::= \alpha\langle e_1, \ldots, e_n\rangle$
- Molecules $M ::= A_1 \mid \cdots \mid A_n$
- Joins $J ::= \alpha_1\langle x_{11}, \ldots, x_{1m_1}\rangle \mid \cdots \mid \alpha_n\langle x_{n1}, \ldots, x_{nm_n}\rangle$
- Reactions $R ::= J \triangleright P$

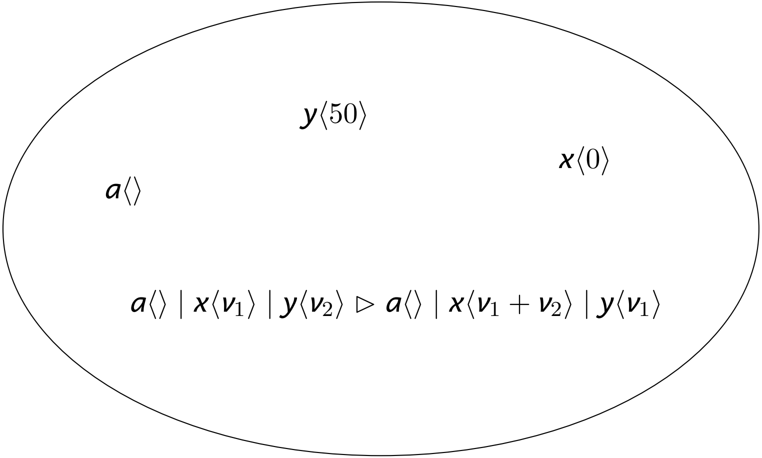
Some context

- Atoms $A ::= \alpha\langle e_1, \ldots, e_n\rangle$
- Molecules $M ::= A_1 \mid \cdots \mid A_n$
- Joins $J ::= \alpha_1\langle x_{11}, \ldots, x_{1m_1}\rangle \mid \cdots \mid \alpha_n\langle x_{n1}, \ldots, x_{nm_n}\rangle$
- Reactions $R ::= J \rhd P$
- Processes $P ::= (R \mid A)^+$ as sets of reactions and atoms

Some context

- Atoms $A ::= \alpha\langle e_1, \dots, e_n\rangle$
- Molecules $M ::= A_1 \mid \cdots \mid A_n$
- Joins $J ::= \alpha_1\langle x_{11}, \dots, x_{1m_1}\rangle \mid \cdots \mid \alpha_n\langle x_{n1}, \dots, x_{nm_n}\rangle$
- Reactions $R ::= J \triangleright P$
- Processes $P ::= (R \mid A)^+$ as sets of reactions and atoms
- Solutions are multisets of molecules and reactions

The Join Calculus, graphically



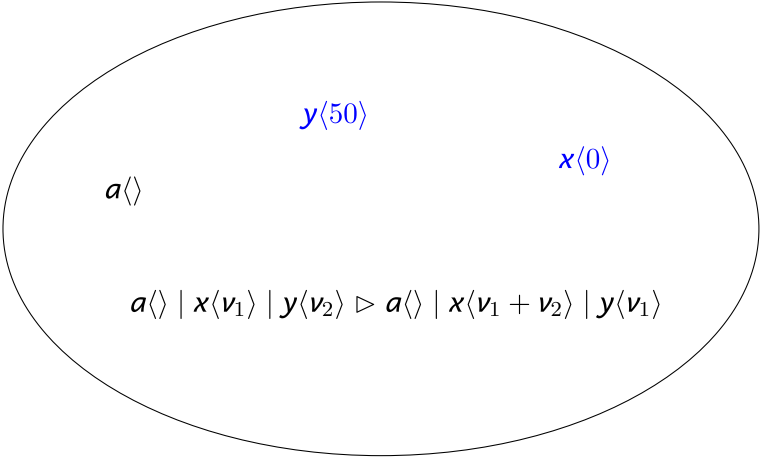
$y\langle 50\rangle$

$x\langle 0\rangle$

$a\langle\rangle$

$a\langle\rangle \mid x\langle v_1\rangle \mid y\langle v_2\rangle \triangleright a\langle\rangle \mid x\langle v_1 + v_2\rangle \mid y\langle v_1\rangle$

Some context



$y\langle 50\rangle$

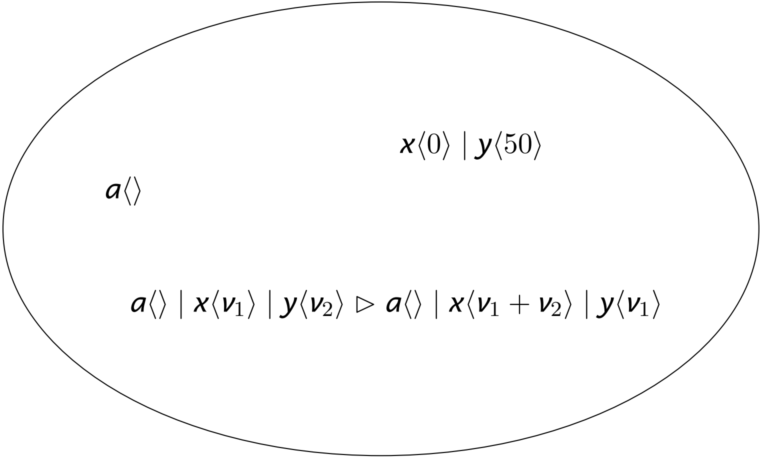
$x\langle 0\rangle$

$a\langle\rangle$

$a\langle\rangle \mid x\langle v_1\rangle \mid y\langle v_2\rangle \triangleright a\langle\rangle \mid x\langle v_1 + v_2\rangle \mid y\langle v_1\rangle$

1. Cooling step $\rightharpoondown$

Some context



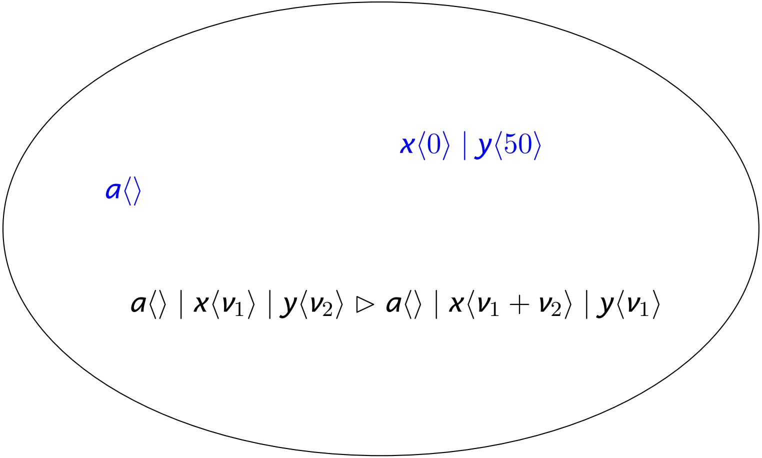
$x\langle 0\rangle \mid y\langle 50\rangle$

$a\langle\rangle$

$a\langle\rangle \mid x\langle v_1\rangle \mid y\langle v_2\rangle \triangleright a\langle\rangle \mid x\langle v_1 + v_2\rangle \mid y\langle v_1\rangle$

1. Cooling step $\rightharpoondown$

Some context



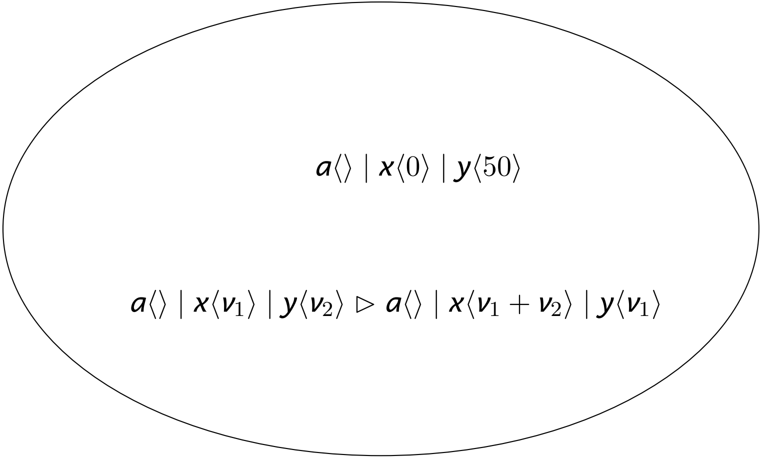
$x\langle 0\rangle \mid y\langle 50\rangle$

$a\langle\rangle$

$a\langle\rangle \mid x\langle v_1\rangle \mid y\langle v_2\rangle \triangleright a\langle\rangle \mid x\langle v_1 + v_2\rangle \mid y\langle v_1\rangle$

1. Cooling step $\rightharpoondown$
2. Cooling step $\rightharpoondown$

The Join Calculus, graphically

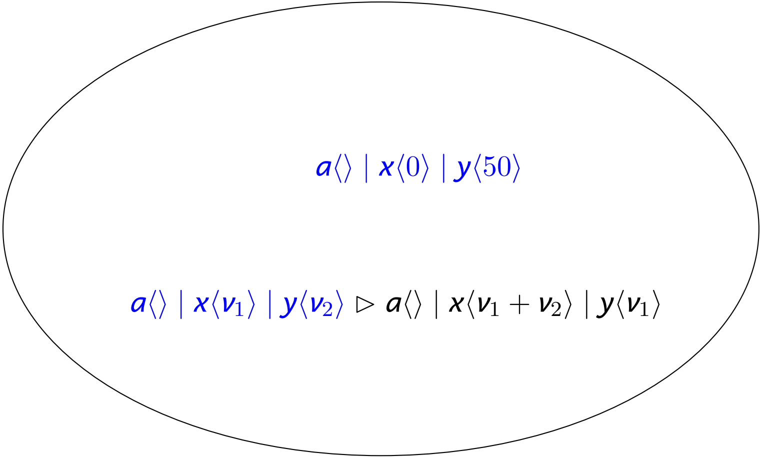


$a\langle\rangle \mid x\langle 0\rangle \mid y\langle 50\rangle$

$a\langle\rangle \mid x\langle v_1\rangle \mid y\langle v_2\rangle \triangleright a\langle\rangle \mid x\langle v_1 + v_2\rangle \mid y\langle v_1\rangle$

1. Cooling step $\rightharpoondown$
2. Cooling step $\rightharpoondown$

The Join Calculus, graphically

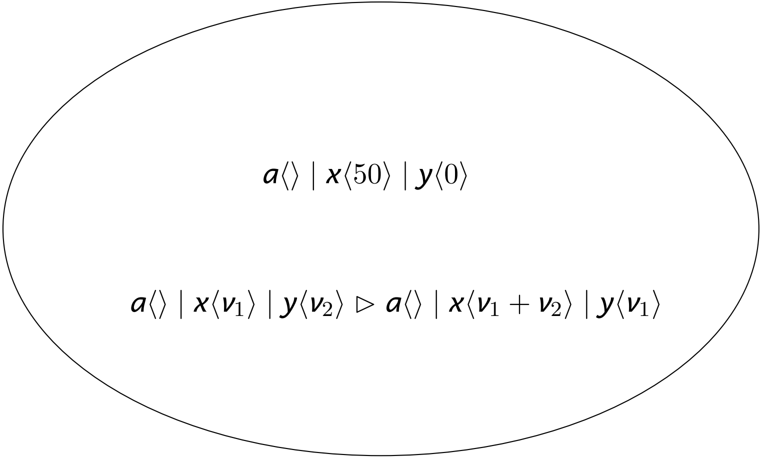


$a\langle\rangle \mid x\langle 0\rangle \mid y\langle 50\rangle$

$a\langle\rangle \mid x\langle v_1\rangle \mid y\langle v_2\rangle \triangleright a\langle\rangle \mid x\langle v_1 + v_2\rangle \mid y\langle v_1\rangle$

1. Cooling step $\rightharpoondown$
2. Cooling step $\rightharpoondown$
3. Reaction step $\rightarrow$ $(v_1 := 0,\ v_2 := 50)$

The Join Calculus, graphically

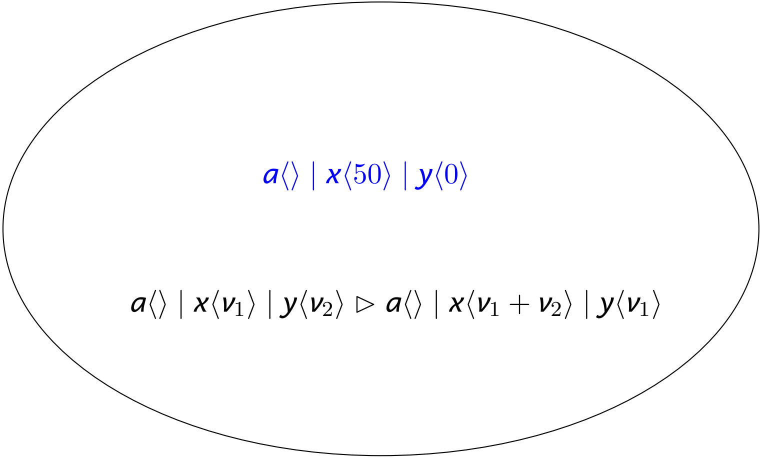


$a\langle\rangle \mid x\langle 50\rangle \mid y\langle 0\rangle$

$a\langle\rangle \mid x\langle v_1\rangle \mid y\langle v_2\rangle \triangleright a\langle\rangle \mid x\langle v_1 + v_2\rangle \mid y\langle v_1\rangle$

1. Cooling step $\rightharpoondown$
2. Cooling step $\rightharpoondown$
3. Reaction step $\rightarrow$ ($v_1 := 0$, $v_2 := 50$)

Some context

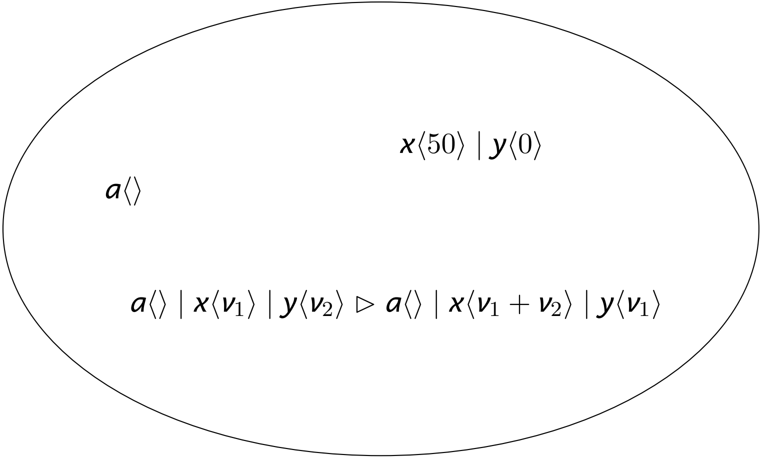


$a\langle\rangle \mid x\langle 50\rangle \mid y\langle 0\rangle$

$a\langle\rangle \mid x\langle v_1\rangle \mid y\langle v_2\rangle \triangleright a\langle\rangle \mid x\langle v_1 + v_2\rangle \mid y\langle v_1\rangle$

1. Cooling step $\rightharpoondown$
2. Cooling step $\rightharpoondown$
3. Reaction step $\rightarrow$ $(v_1 := 0,\ v_2 := 50)$
4. Heating step $\rightharpoonup$

The Join Calculus, graphically



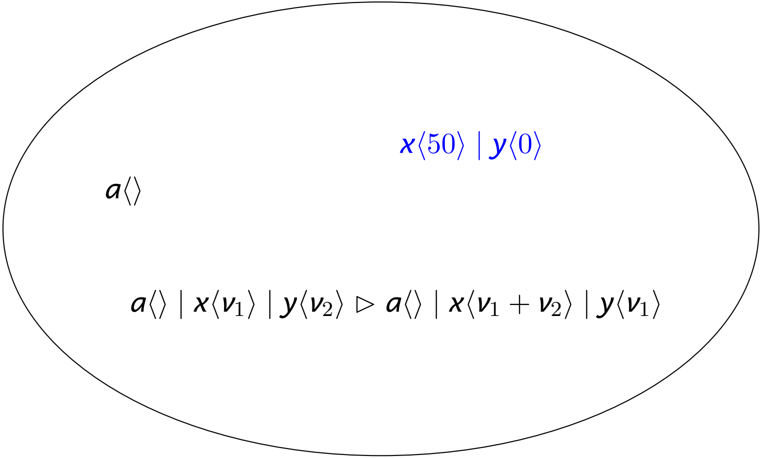
$x\langle 50\rangle \mid y\langle 0\rangle$

$a\langle\rangle$

$a\langle\rangle \mid x\langle v_1\rangle \mid y\langle v_2\rangle \triangleright a\langle\rangle \mid x\langle v_1 + v_2\rangle \mid y\langle v_1\rangle$

1. Cooling step $\rightharpoondown$
2. Cooling step $\rightharpoondown$
3. Reaction step $\rightarrow$ ($v_1 := 0$, $v_2 := 50$)
4. Heating step $\rightharpoonup$

Some context



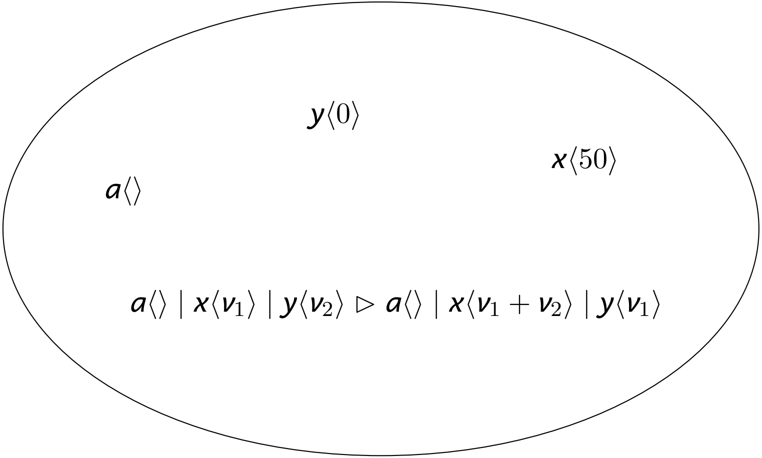
$x\langle 50\rangle \mid y\langle 0\rangle$

$a\langle\rangle$

$a\langle\rangle \mid x\langle v_1\rangle \mid y\langle v_2\rangle \triangleright a\langle\rangle \mid x\langle v_1 + v_2\rangle \mid y\langle v_1\rangle$

1. Cooling step $\rightharpoondown$
2. Cooling step $\rightharpoondown$
3. Reaction step $\rightarrow$ ($v_1 := 0,\ v_2 := 50$)
4. Heating step $\rightharpoonup$
5. Heating step $\rightharpoonup$

The Join Calculus, graphically



$y\langle 0\rangle$

$x\langle 50\rangle$

$a\langle\rangle$

$a\langle\rangle \mid x\langle v_1\rangle \mid y\langle v_2\rangle \triangleright a\langle\rangle \mid x\langle v_1 + v_2\rangle \mid y\langle v_1\rangle$

1. Cooling step $\rightharpoondown$
2. Cooling step $\rightharpoondown$
3. Reaction step $\rightarrow$ ($v_1 := 0$, $v_2 := 50$)
4. Heating step $\rightharpoonup$
5. Heating step $\rightharpoonup$

Encoding Distributed PlusCal into the Join Calculus

- Each process is its own subsolution

Encoding Distributed PlusCal into the Join Calculus

- Each process is its own subsolution
- Encode each variable x as an atom $x\langle v\rangle$

Encoding Distributed PlusCal into the Join Calculus

- Each process is its own subsolution
- Encode each variable x as an atom $x\langle v\rangle$
- Encode channels c as multiple atoms $c\langle v\rangle$ for each message v

Encoding Distributed PlusCal into the Join Calculus

- Each process is its own subsolution
- Encode each variable x as an atom $x\langle v \rangle$
- Encode channels c as multiple atoms $c\langle v \rangle$ for each message v
- For each atomic block, define a reaction which

Encoding Distributed PlusCal into the Join Calculus

- Each process is its own subsolution
- Encode each variable x as an atom $x\langle v\rangle$
- Encode channels c as multiple atoms $c\langle v\rangle$ for each message v
- For each atomic block, define a reaction which
  - consumes the message associated to the label,

Encoding Distributed PlusCal into the Join Calculus

- Each process is its own subsolution
- Encode each variable x as an atom $x\langle v\rangle$
- Encode channels c as multiple atoms $c\langle v\rangle$ for each message v
- For each atomic block, define a reaction which
  - consumes the message associated to the label,
  - joins all variables/channels used,

Encoding Distributed PlusCal into the Join Calculus

- Each process is its own subsolution
- Encode each variable x as an atom $x\langle v\rangle$
- Encode channels c as multiple atoms $c\langle v\rangle$ for each message v
- For each atomic block, define a reaction which
  - consumes the message associated to the label,
  - joins all variables/channels used,
  - is guarded by all `await` conditions

Encoding Distributed PlusCal into the Join Calculus

- Each process is its own subsolution
- Encode each variable x as an atom $x\langle v\rangle$
- Encode channels c as multiple atoms $c\langle v\rangle$ for each message v
- For each atomic block, define a reaction which
  - consumes the message associated to the label,
  - joins all variables/channels used,
  - is guarded by all `await` conditions
  - emits new variables/channels and the next label

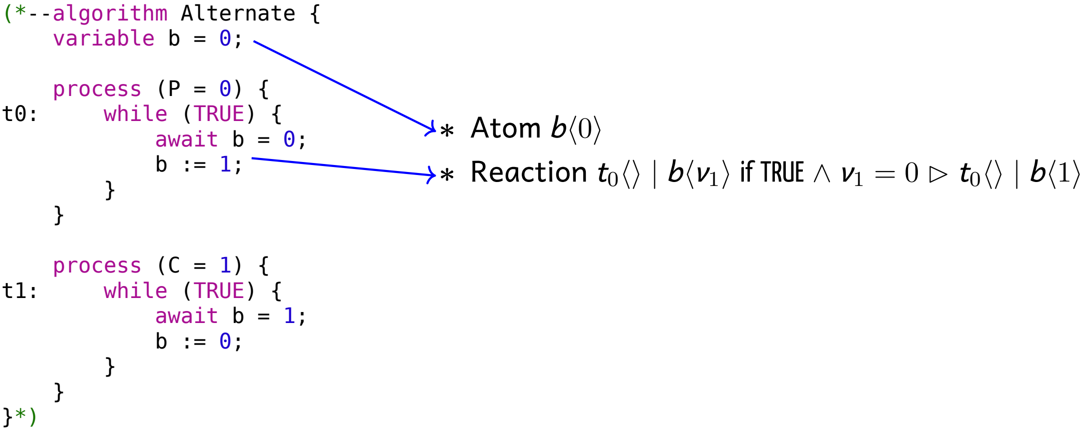
```
(*--algorithm Alternate {
    variable b = 0;

    process (P = 0) {
t0:     while (TRUE) {
            await b = 0;
            b := 1;
        }
    }

    process (C = 1) {
t1:     while (TRUE) {
            await b = 1;
            b := 0;
        }
    }
}*)
```

Encoding Distributed PlusCal into the Join Calculus

```
(*--algorithm Alternate {
    variable b = 0;

    process (P = 0) {
t0:     while (TRUE) {
            await b = 0;
            b := 1;
        }
    }

    process (C = 1) {
t1:     while (TRUE) {
            await b = 1;
            b := 0;
        }
    }
}*)
```

- Atom $b\langle 0 \rangle$

Encoding Distributed PlusCal into the Join Calculus



```
(*--algorithm Alternate {
    variable b = 0;

    process (P = 0) {
t0:     while (TRUE) {
            await b = 0;
            b := 1;
        }
    }

    process (C = 1) {
t1:     while (TRUE) {
            await b = 1;
            b := 0;
        }
    }
}*)
```

- Atom $b\langle 0\rangle$
- Reaction $t_0\langle\rangle \mid b\langle v_1\rangle$ if TRUE $\wedge\ v_1 = 0 \triangleright t_0\langle\rangle \mid b\langle 1\rangle$

Encoding Distributed PlusCal into the Join Calculus

Short example

```
(*--algorithm Alternate {
    variable b = 0;

    process (P = 0) {
t0:     while (TRUE) {
            await b = 0;
            b := 1;
        }
    }

    process (C = 1) {
t1:     while (TRUE) {
            await b = 1;
            b := 0;
        }
    }
}*)
```

- Atom $b\langle 0\rangle$
- Reaction $t_0\langle\rangle \mid b\langle v_1\rangle$ if TRUE $\land\ v_1 = 0 \triangleright t_0\langle\rangle \mid b\langle 1\rangle$

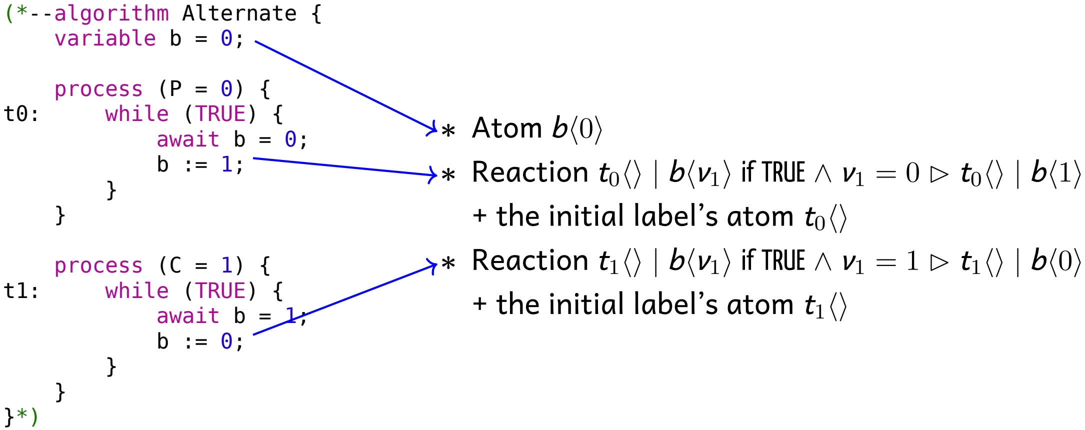
+ the initial label's atom $t_0\langle\rangle$

```
(*--algorithm Alternate {
    variable b = 0;

    process (P = 0) {
t0:     while (TRUE) {
            await b = 0;
            b := 1;
        }
    }

    process (C = 1) {
t1:     while (TRUE) {
            await b = 1;
            b := 0;
        }
    }
}*)
```

- Atom $b\langle 0\rangle$
- Reaction $t_0\langle\rangle \mid b\langle v_1\rangle$ if TRUE $\land\ v_1 = 0 \triangleright t_0\langle\rangle \mid b\langle 1\rangle$

  + the initial label's atom $t_0\langle\rangle$
- Reaction $t_1\langle\rangle \mid b\langle v_1\rangle$ if TRUE $\land\ v_1 = 1 \triangleright t_1\langle\rangle \mid b\langle 0\rangle$

Encoding Distributed PlusCal into the Join Calculus



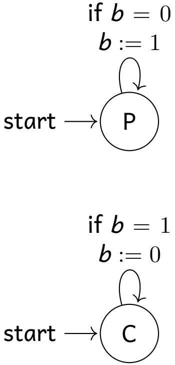
```
(*--algorithm Alternate {
    variable b = 0;

    process (P = 0) {
t0:     while (TRUE) {
            await b = 0;
            b := 1;
        }
    }

    process (C = 1) {
t1:     while (TRUE) {
            await b = 1;
            b := 0;
        }
    }
}*)
```

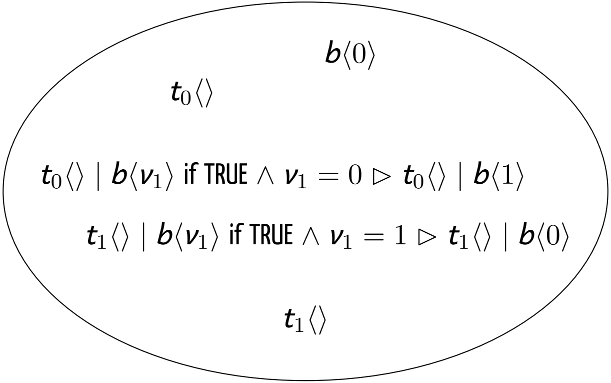
- $*$ Atom $b\langle 0\rangle$
- $*$ Reaction $t_0\langle\rangle \mid b\langle v_1\rangle$ if TRUE $\wedge\ v_1 = 0 \triangleright t_0\langle\rangle \mid b\langle 1\rangle$

  + the initial label's atom $t_0\langle\rangle$
- $*$ Reaction $t_1\langle\rangle \mid b\langle v_1\rangle$ if TRUE $\wedge\ v_1 = 1 \triangleright t_1\langle\rangle \mid b\langle 0\rangle$

  + the initial label's atom $t_1\langle\rangle$

Encoding Distributed PlusCal into the Join Calculus





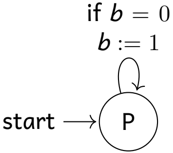
$b\langle 0\rangle$

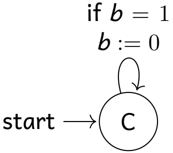
$t_0\langle\rangle$

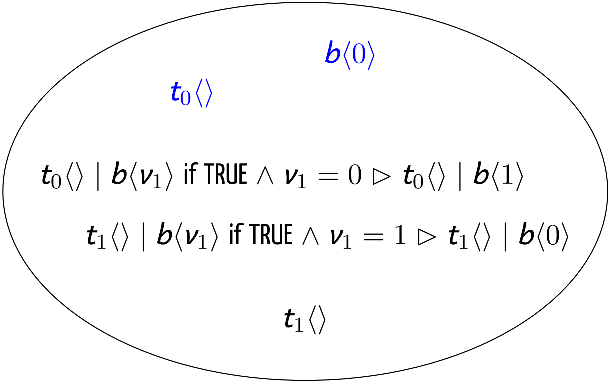
$t_0\langle\rangle \mid b\langle v_1\rangle$ if TRUE $\wedge\ v_1 = 0 \triangleright t_0\langle\rangle \mid b\langle 1\rangle$

$t_1\langle\rangle \mid b\langle v_1\rangle$ if TRUE $\wedge\ v_1 = 1 \triangleright t_1\langle\rangle \mid b\langle 0\rangle$

$t_1\langle\rangle$







$b\langle 0\rangle$

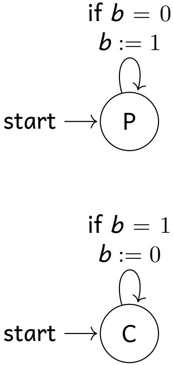
$t_0\langle\rangle$

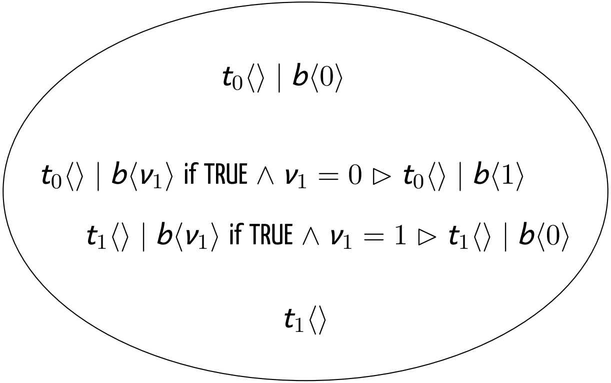
$t_0\langle\rangle \mid b\langle v_1\rangle \text{ if TRUE} \wedge v_1 = 0 \triangleright t_0\langle\rangle \mid b\langle 1\rangle$

$t_1\langle\rangle \mid b\langle v_1\rangle \text{ if TRUE} \wedge v_1 = 1 \triangleright t_1\langle\rangle \mid b\langle 0\rangle$

$t_1\langle\rangle$

Comparison with the initial automata



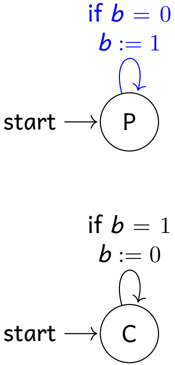


$$t_0\langle\rangle \mid b\langle 0\rangle$$

$$t_0\langle\rangle \mid b\langle v_1\rangle \text{ if TRUE} \wedge v_1 = 0 \rhd t_0\langle\rangle \mid b\langle 1\rangle$$

$$t_1\langle\rangle \mid b\langle v_1\rangle \text{ if TRUE} \wedge v_1 = 1 \rhd t_1\langle\rangle \mid b\langle 0\rangle$$

$$t_1\langle\rangle$$



$t_0\langle\rangle \mid b\langle 0\rangle$

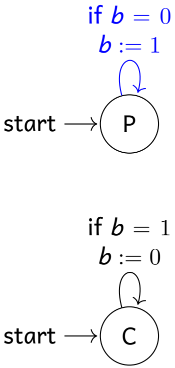
$t_0\langle\rangle \mid b\langle v_1\rangle \text{ if TRUE} \wedge v_1 = 0 \triangleright t_0\langle\rangle \mid b\langle 1\rangle$

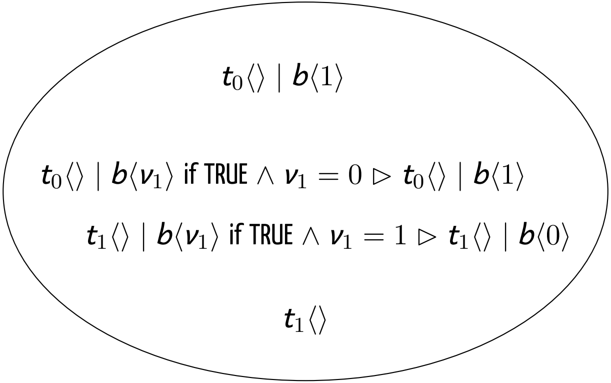
$t_1\langle\rangle \mid b\langle v_1\rangle \text{ if TRUE} \wedge v_1 = 1 \triangleright t_1\langle\rangle \mid b\langle 0\rangle$

$t_1\langle\rangle$

$\rightarrow (v_1 := 0)$

Comparison with the initial automata





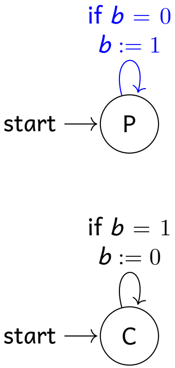
$t_0\langle\rangle \mid b\langle 1\rangle$

$t_0\langle\rangle \mid b\langle v_1\rangle \text{ if TRUE} \wedge v_1 = 0 \triangleright t_0\langle\rangle \mid b\langle 1\rangle$

$t_1\langle\rangle \mid b\langle v_1\rangle \text{ if TRUE} \wedge v_1 = 1 \triangleright t_1\langle\rangle \mid b\langle 0\rangle$

$t_1\langle\rangle$

Comparison with the initial automata



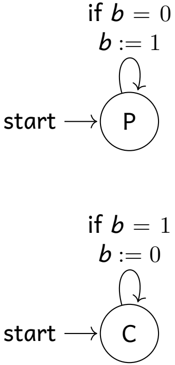
$t_0\langle\rangle \mid b\langle 1\rangle$

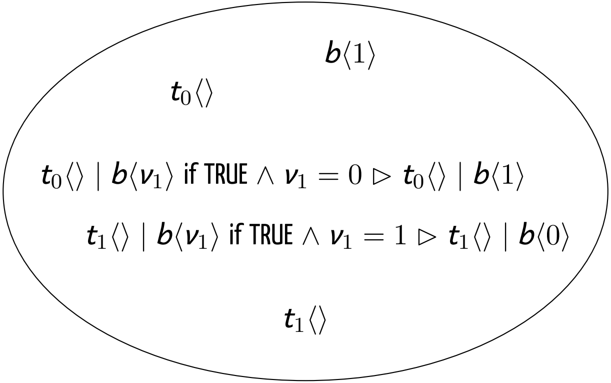
$t_0\langle\rangle \mid b\langle v_1\rangle$ if TRUE $\wedge\ v_1 = 0 \rhd t_0\langle\rangle \mid b\langle 1\rangle$

$t_1\langle\rangle \mid b\langle v_1\rangle$ if TRUE $\wedge\ v_1 = 1 \rhd t_1\langle\rangle \mid b\langle 0\rangle$

$t_1\langle\rangle$

$\longrightarrow$





$b\langle 1\rangle$

$t_0\langle\rangle$

$t_0\langle\rangle \mid b\langle v_1\rangle$ if TRUE $\wedge\ v_1 = 0 \triangleright t_0\langle\rangle \mid b\langle 1\rangle$

$t_1\langle\rangle \mid b\langle v_1\rangle$ if TRUE $\wedge\ v_1 = 1 \triangleright t_1\langle\rangle \mid b\langle 0\rangle$

$t_1\langle\rangle$

Problems yet to be solved

- How to efficiently encode guarded reactions in the Join Calculus?

Problems yet to be solved

- How to efficiently encode guarded reactions in the Join Calculus?
- (Mechanical) proof of semantics preservation...

Problems yet to be solved

- How to efficiently encode guarded reactions in the Join Calculus?
- (Mechanical) proof of semantics preservation...
    - ...from Distributed PlusCal to the Join Calculus?

Problems yet to be solved

- How to efficiently encode guarded reactions in the Join Calculus?
- (Mechanical) proof of semantics preservation...
    - ...from Distributed PlusCal to the Join Calculus?
    - ...from the Join Calculus to executable code?