# Towards Practical Verification of Protocols in mCRL2*

*Ongoing* work by
**Bas van den Heuvel** and Jorge A. Pérez

Bernoulli Institute for Math, CS, and AI
University of Groningen, The Netherlands

AGERE 2021

*Research partially supported by the Dutch Research Council (NWO) under project No. 016.Vidi.189.046 (Unifying Correctness for Communicating Software).

# Overview

- Context:

# Overview

- Context:
  - Multiparty Session Types (MPST),

# Overview

- Context:
  - Multiparty Session Types (MPST),
  - Traditional method of verification:
    projection onto *local protocols* for type checking,

# Overview

- Context:
  - Multiparty Session Types (MPST),
  - Traditional method of verification:
    projection onto *local protocols* for type checking,
  - Our novel approach:
    verification by means of *model checking process implementations*.

# Overview

- Context:
  - Multiparty Session Types (MPST),
  - Traditional method of verification:
    projection onto *local protocols* for type checking,
  - Our novel approach:
    verification by means of *model checking process implementations*.
- Implementing MPST as communicating processes.

# Overview

- Context:
  - Multiparty Session Types (MPST),
  - Traditional method of verification:
    projection onto *local protocols* for type checking,
  - Our novel approach:
    verification by means of *model checking process implementations*.
- Implementing MPST as communicating processes.
- Current work: formulas for model checking.

# Context: Multiparty Session Types (MPST)

- MPST: theory of protocols that describe interaction between two or more participants *from a vantage point*.

# Context: Multiparty Session Types (MPST)

- MPST: theory of protocols that describe interaction between two or more participants *from a vantage point*.
- Usually expressed as *global types*:

$$
\begin{aligned}
G ::= {} & s \twoheadrightarrow r\langle T \rangle \,.\, G && T \text{ is data type, e.g. “int” or “str”} \\
\mid {} & s \twoheadrightarrow r(\ell_i \,.\, G)_{\ell_i \in I} && \ell_i \text{ are labels, e.g. “heads” or “tails”} \\
\mid {} & \mathsf{end} \\
\mid {} & \mathsf{rec}\, X \,.\, G \mid X && X \text{ must be guarded by exchange in } G
\end{aligned}
$$

# Context: Multiparty Session Types (MPST)

- MPST: theory of protocols that describe interaction between two or more participants *from a vantage point*.
- Usually expressed as *global types*:

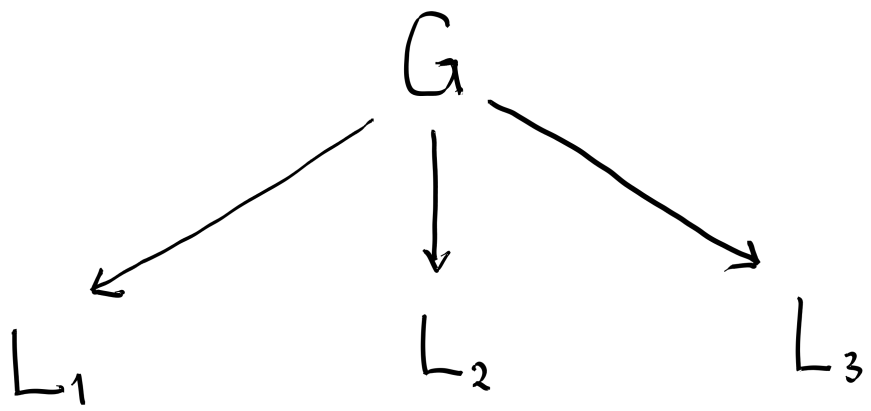
$$
\begin{aligned}
G ::= {} & s \twoheadrightarrow r\langle T \rangle \,.\, G && T \text{ is data type, e.g. “int” or “str”} \\
\mid {} & s \twoheadrightarrow r(\ell_i \,.\, G)_{\ell_i \in I} && \ell_i \text{ are labels, e.g. “heads” or “tails”} \\
\mid {} & \mathsf{end} \\
\mid {} & \mathsf{rec}\, X \,.\, G \mid X && X \text{ must be guarded by exchange in } G
\end{aligned}
$$

- Running example global type with participants Alice ($a$), Bob ($b$), and Coin ($c$):

$$
G_{\mathsf{coin}} := \mathsf{rec}\, X \,.\, a \twoheadrightarrow b\langle \mathsf{nat} \rangle \,.\, a \twoheadrightarrow b \left( \begin{array}{l} \mathsf{heads}.c \twoheadrightarrow b \begin{pmatrix} \mathsf{heads}.b \twoheadrightarrow a(\mathsf{win}.X), \\ \mathsf{tails}.b \twoheadrightarrow a(\mathsf{lose}.\mathsf{end}) \end{pmatrix}, \\ \mathsf{tails}.c \twoheadrightarrow b \begin{pmatrix} \mathsf{heads}.b \twoheadrightarrow a(\mathsf{lose}.\mathsf{end}), \\ \mathsf{tails}.b \twoheadrightarrow a(\mathsf{win}.X) \end{pmatrix} \end{array} \right)
$$

# Context: Traditional methods of verification

- Global types are projected onto participants yielding *local types*.



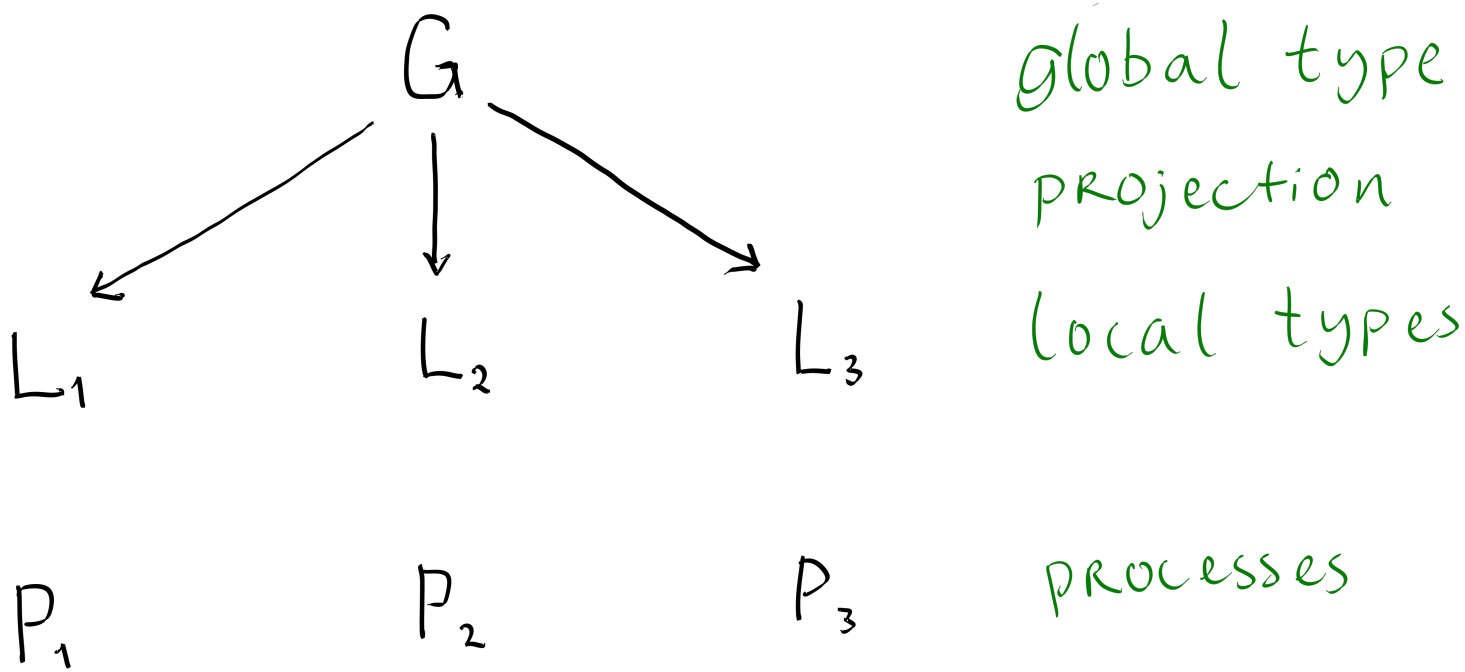

global type

projection

local types

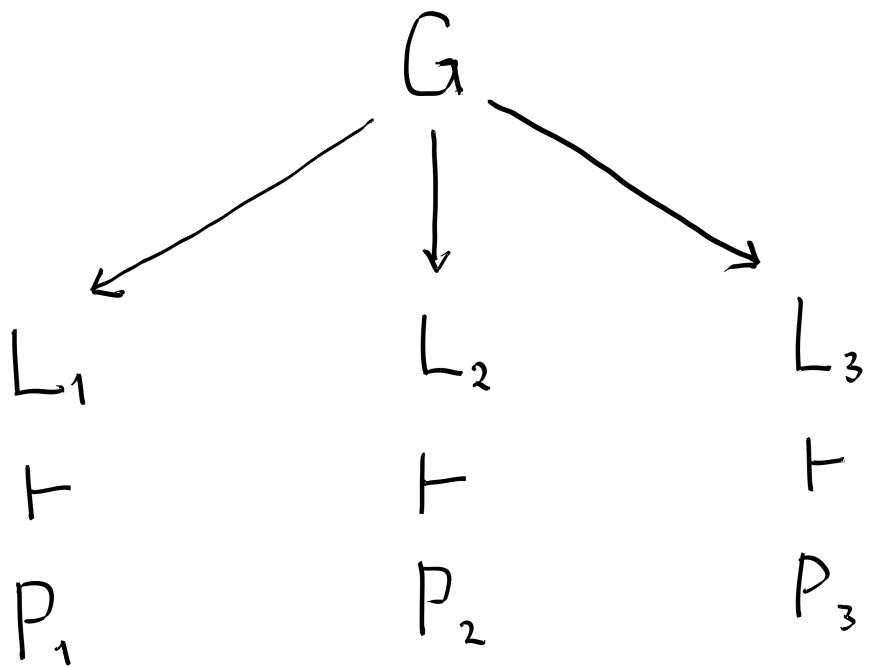
# Context: Traditional methods of verification

- Global types are projected onto participants yielding *local types*.
- Roles of participants are implemented as processes (usually $\pi$-calculus).



# Context: Traditional methods of verification

- Global types are projected onto participants yielding *local types*.
- Roles of participants are implemented as processes (usually $\pi$-calculus).
- Process implementations are type checked using the local types.




global type
projection
local types
type checking
processes

# Context: Our approach

- Alternative method of verification: *model checking*.

# Context: Our approach

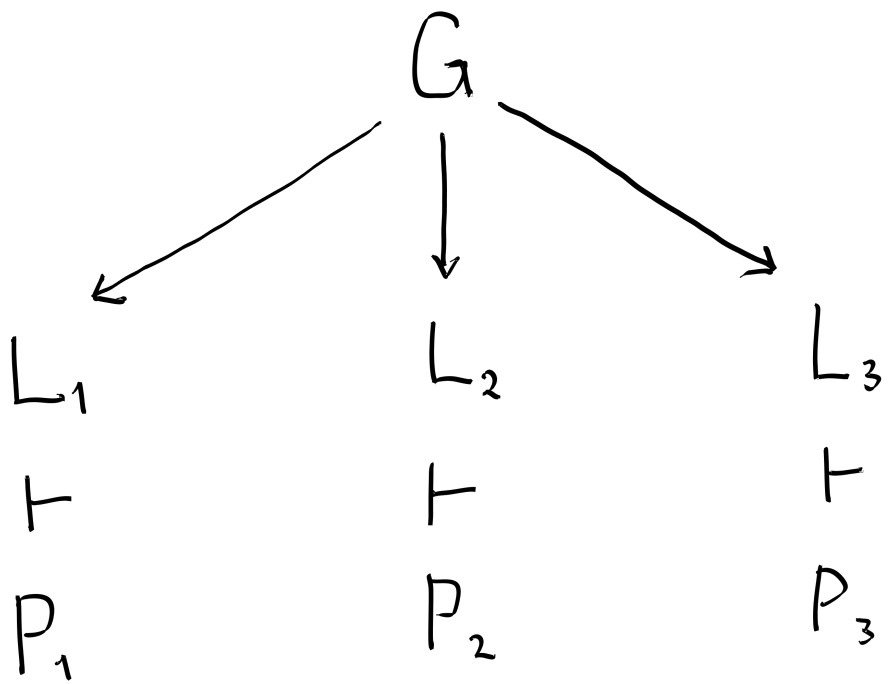
- Alternative method of verification: *model checking*.
- Prior works apply model checking only to *types*.

# Context: Our approach

- Alternative method of verification: *model checking*.
- Prior works apply model checking only to *types*.
- Our novel approach focusses on model checking *implementations as communicating processes*.

# Context: Our approach

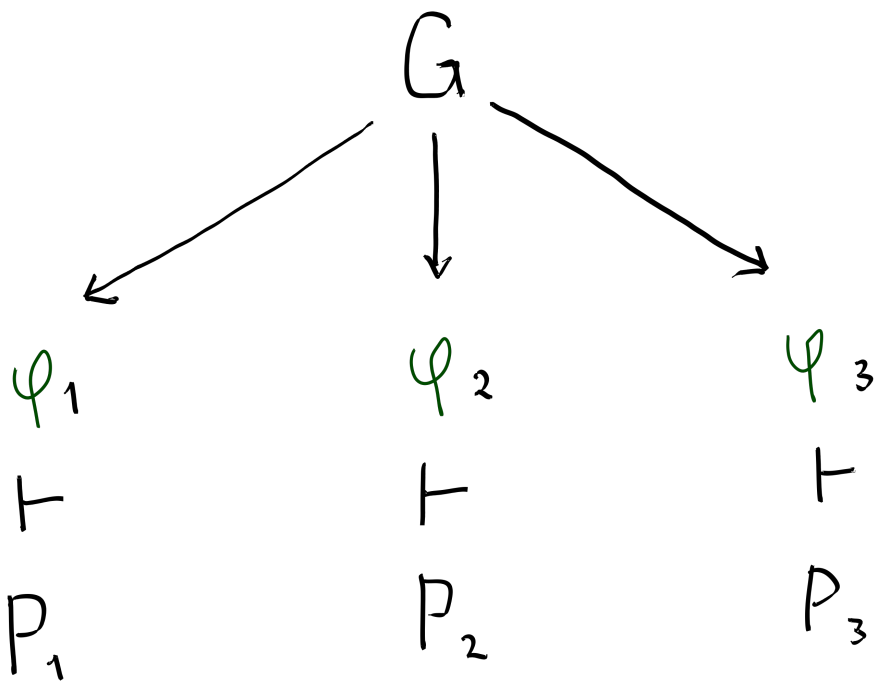
- Alternative method of verification: *model checking*.
- Prior works apply model checking only to *types*.
- Our novel approach focusses on model checking *implementations as communicating processes*.




global type
projection
local types
type checking
processes

# Context: Our approach

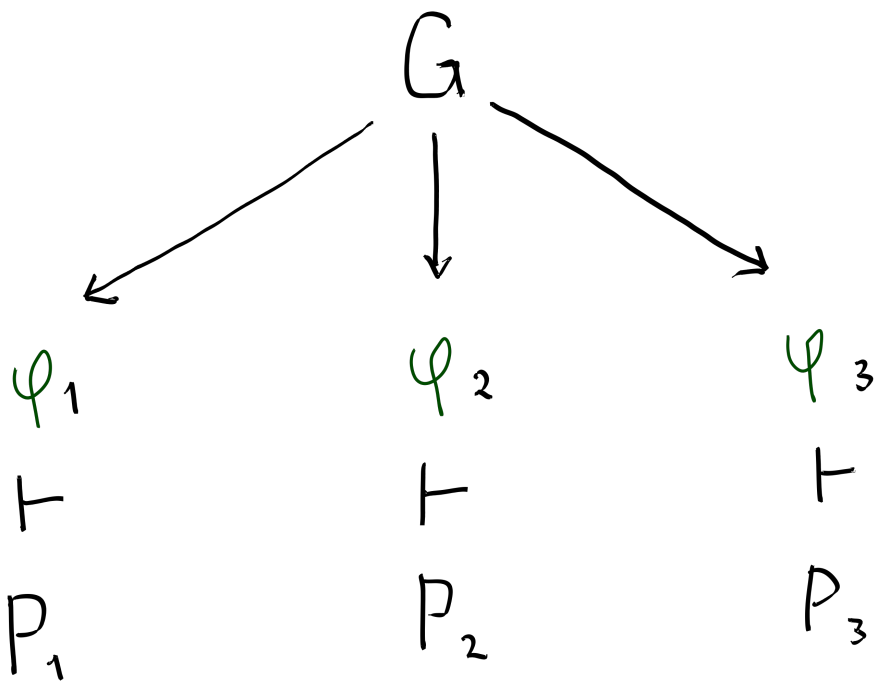
- Alternative method of verification: *model checking*.
- Prior works apply model checking only to *types*.
- Our novel approach focusses on model checking *implementations as communicating processes*.




global type
projection
local formulas
type checking
processes

# Context: Our approach

- Alternative method of verification: *model checking*.
- Prior works apply model checking only to *types*.
- Our novel approach focusses on model checking *implementations as communicating processes*.

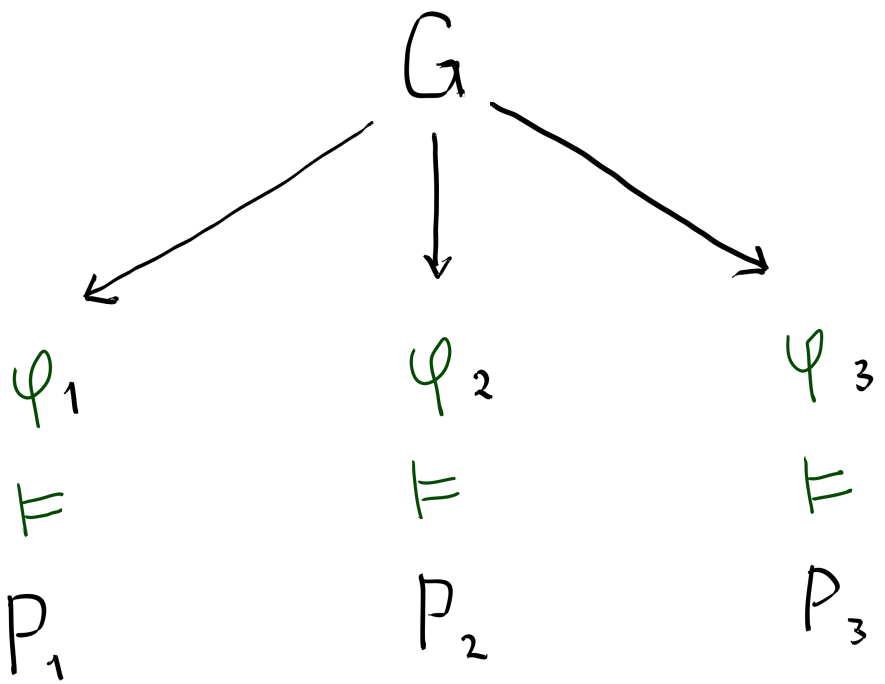



global type derivation

local formulas type checking processes

# Context: Our approach

- Alternative method of verification: *model checking*.
- Prior works apply model checking only to *types*.
- Our novel approach focusses on model checking *implementations as communicating processes*.




global type derivation

local formulas

model checking

processes

# Context: Our approach

- Target: the mCRL2 model checker, based on the Algebra of Communicating Processes and the modal $\mu$-calculus.

# Context: Our approach

- Target: the mCRL2 model checker, based on the Algebra of Communicating Processes and the modal $\mu$-calculus.
- Implement the role of each of a global type's participants *as a separate process*: participant implementations.

# Context: Our approach

- Target: the mCRL2 model checker, based on the Algebra of Communicating Processes and the modal $\mu$-calculus.
- Implement the role of each of a global type's participants *as a separate process*: participant implementations.
- Composition of participant implementations: global implementation.

# Context: Our approach

- Target: the mCRL2 model checker, based on the Algebra of Communicating Processes and the modal $\mu$-calculus.
- Implement the role of each of a global type's participants *as a separate process*: participant implementations.
- Composition of participant implementations: global implementation.
- Derive from global type:
  *local formulas* to model check participant implementations,
  *global formula* to model check global implementation.

# Context: Our approach

- Target: the mCRL2 model checker, based on the Algebra of Communicating Processes and the modal $\mu$-calculus.
- Implement the role of each of a global type's participants *as a separate process*: participant implementations.
- Composition of participant implementations: global implementation.
- Derive from global type:
  *local formulas* to model check participant implementations,
  *global formula* to model check global implementation.
- Ultimate goal:
  compiler from global type specifications to mCRL2 projects.

# Context: Our approach

- Intended workflow:

# Context: Our approach

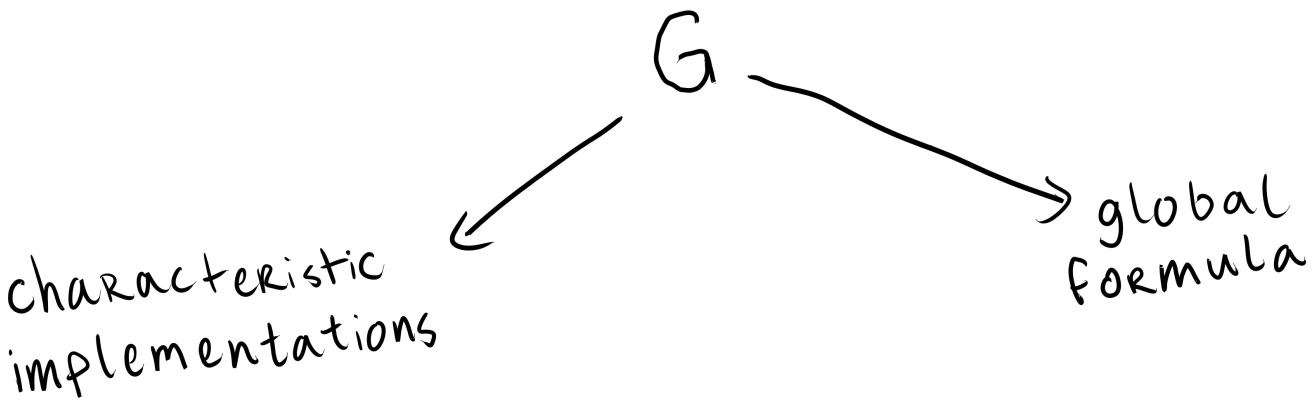
- Intended workflow:

# Context: Our approach

- Intended workflow:

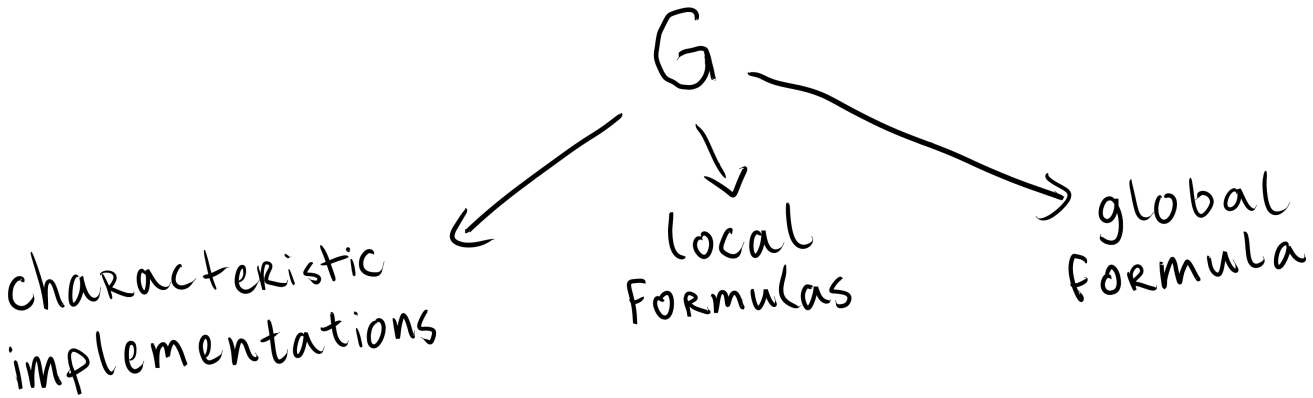
# Context: Our approach

- Intended workflow:



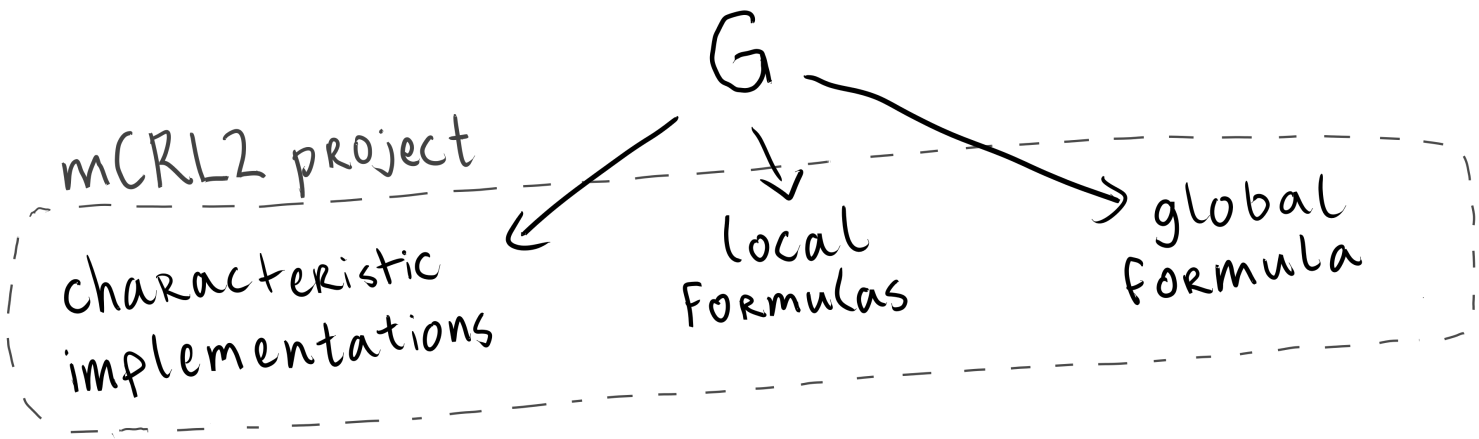
# Context: Our approach

- Intended workflow:



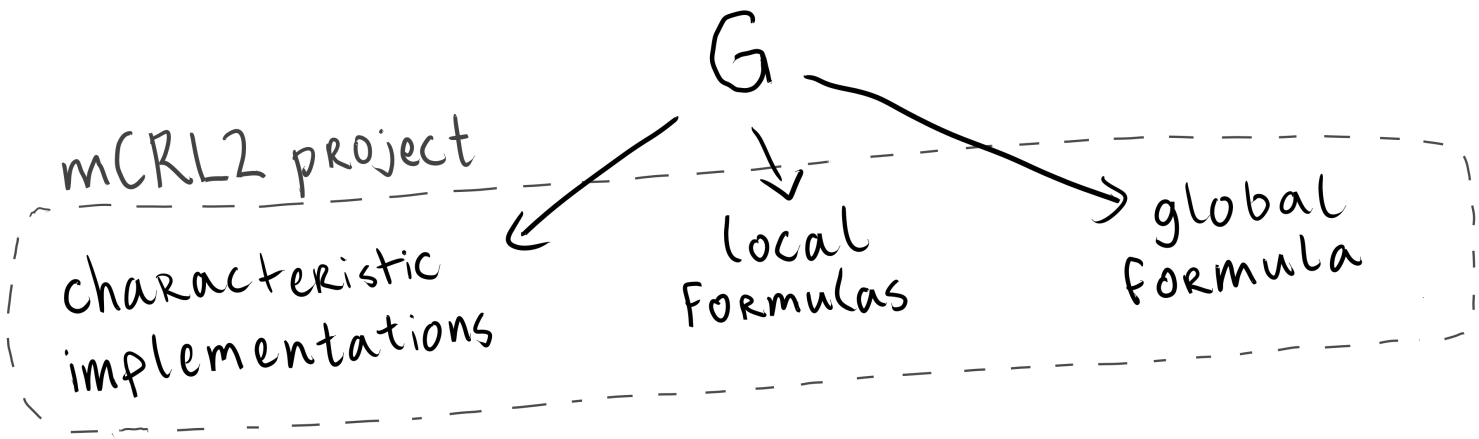
# Context: Our approach

- Intended workflow:



# Context: Our approach

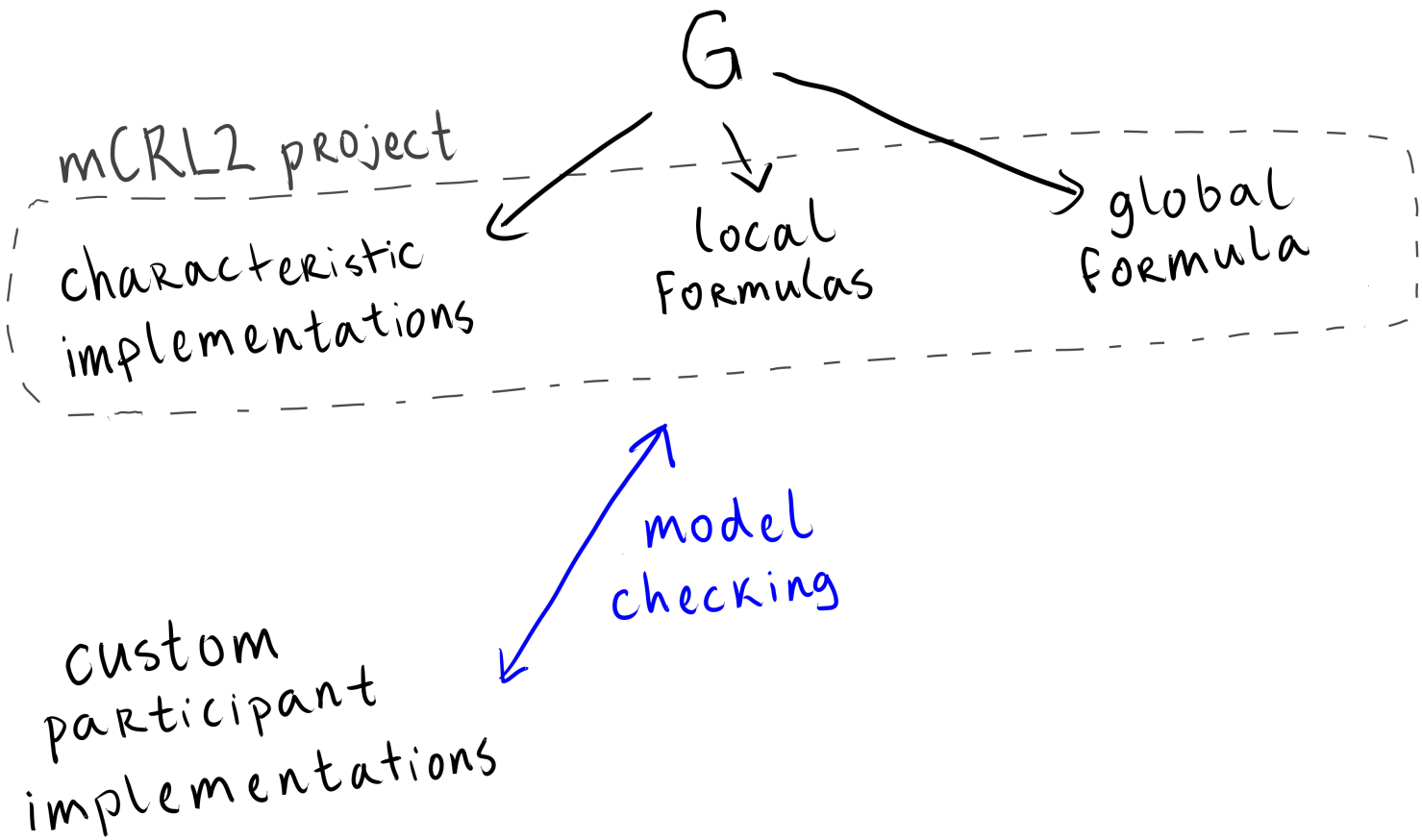
- Intended workflow:




custom participant implementations

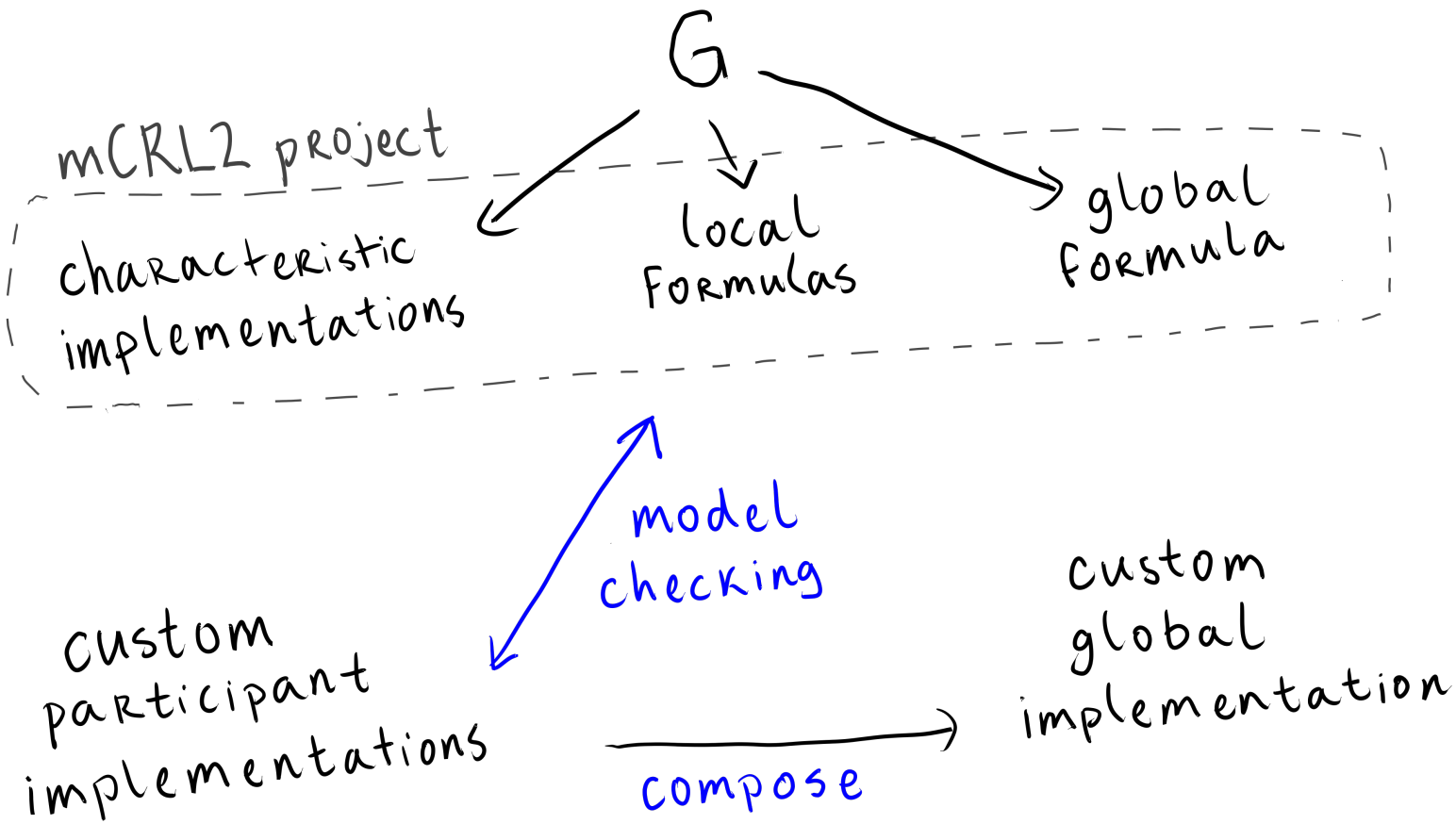
# Context: Our approach

- Intended workflow:



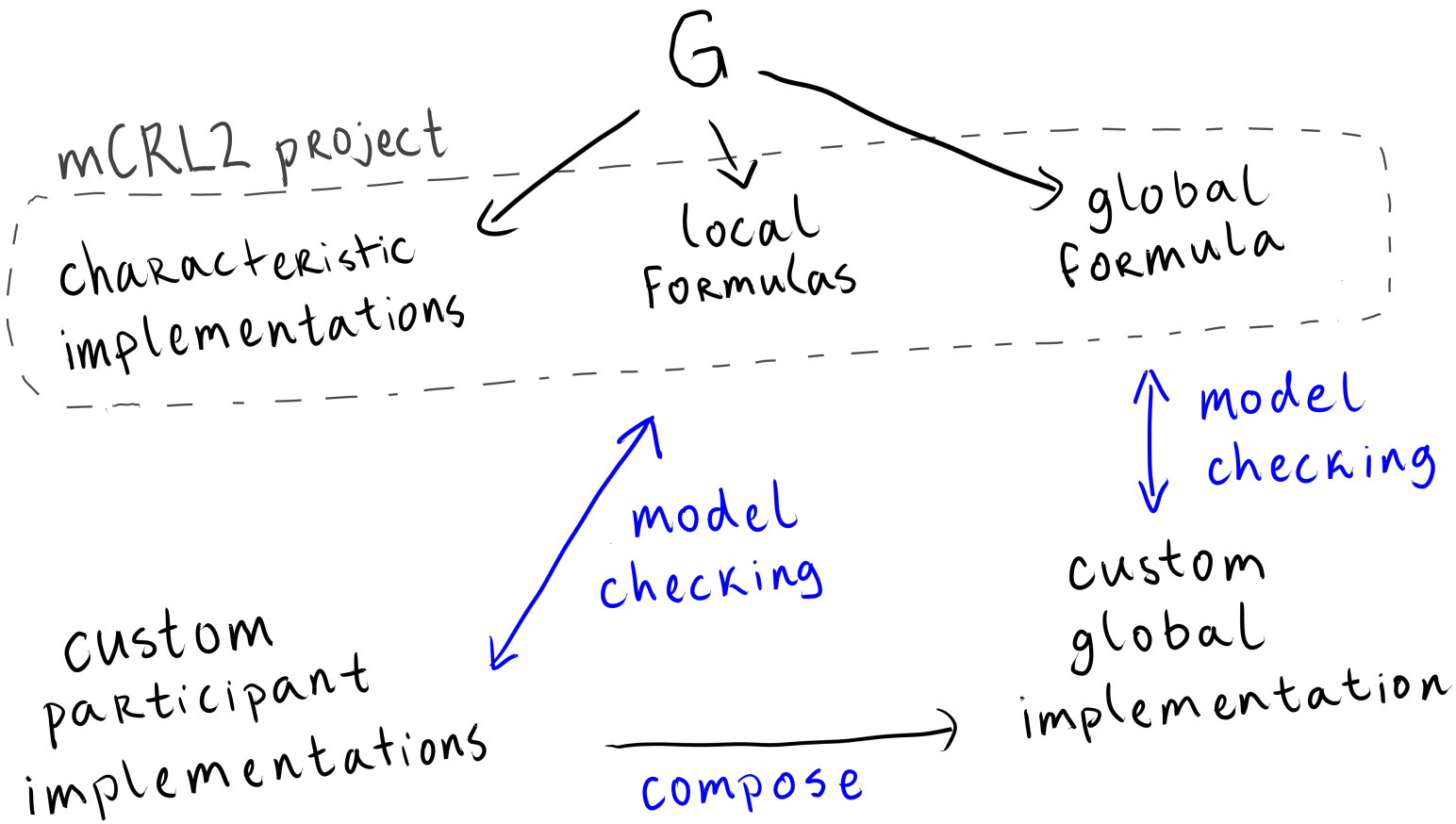
# Context: Our approach

- Intended workflow:



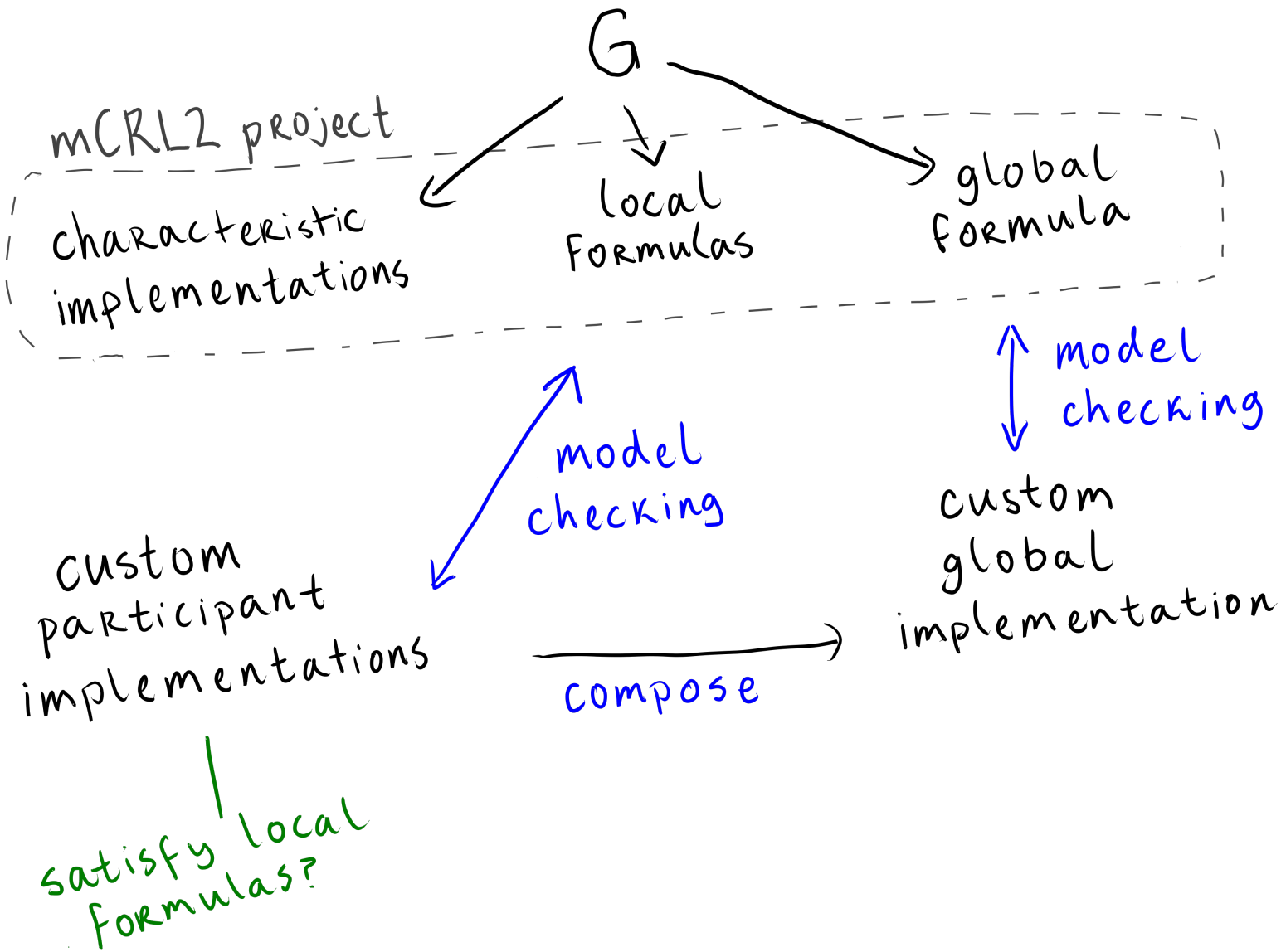
# Context: Our approach

- Intended workflow:



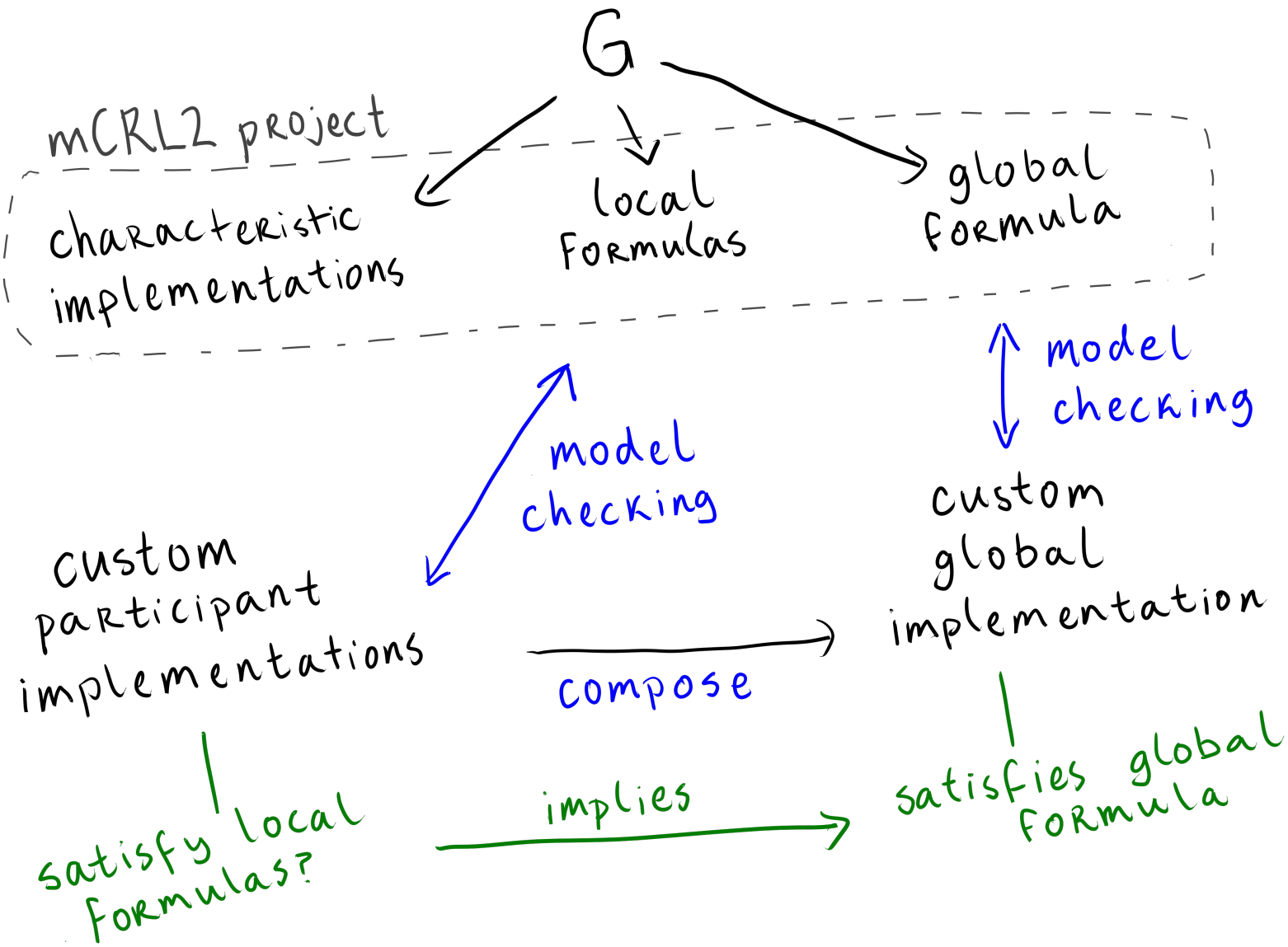
# Context: Our approach

- Intended workflow:



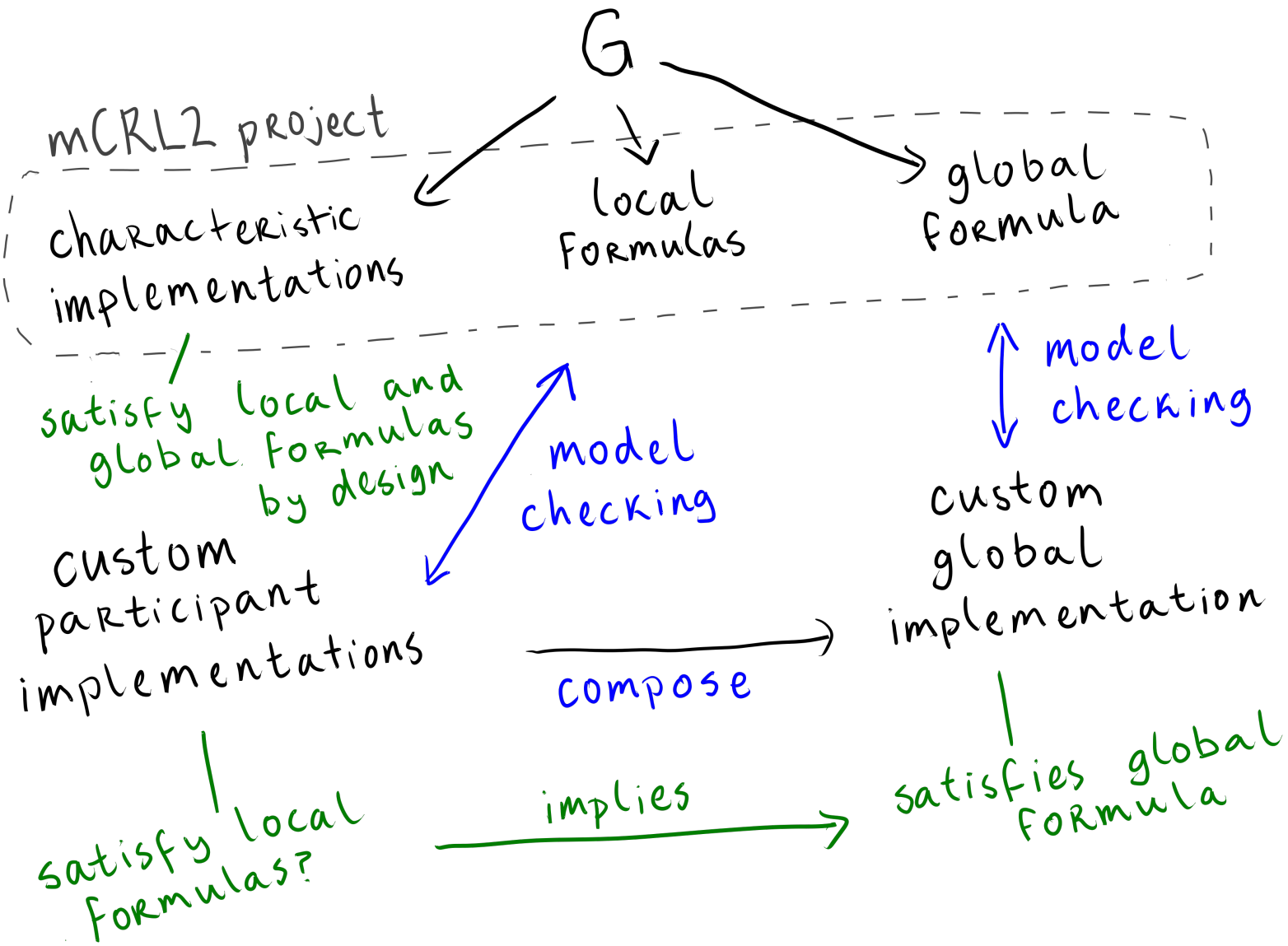
# Context: Our approach

- Intended workflow:



# Context: Our approach

- Intended workflow:



# Implementing MPST as communicating processes

$$G_{\text{coin}} := \text{rec}\, X\,.\, a \twoheadrightarrow b\langle \text{nat} \rangle\,.\, a \twoheadrightarrow b \left( \begin{array}{l} \text{heads}\,.\, c \twoheadrightarrow b \begin{pmatrix} \text{heads}\,.\, b \twoheadrightarrow a(\text{win}\,.\, X), \\ \text{tails}\,.\, b \twoheadrightarrow a(\text{lose}\,.\, \text{end}) \end{pmatrix}, \\ \\ \text{tails}\,.\, c \twoheadrightarrow b \begin{pmatrix} \text{heads}\,.\, b \twoheadrightarrow a(\text{lose}\,.\, \text{end}), \\ \text{tails}\,.\, b \twoheadrightarrow a(\text{win}\,.\, X) \end{pmatrix} \end{array} \right)$$

# Implementing MPST as communicating processes

- Participant implementations consist of *protocol actions*, denoting the input and output of values and labels. E.g., $a[\text{heads}]$ and $b(\text{heads})$.

$$G_{\text{coin}} := \text{rec}\, X\,.\, a \twoheadrightarrow b\langle \text{nat} \rangle\,.\, a \twoheadrightarrow b \left( \begin{array}{l} \text{heads}.c \twoheadrightarrow b \left( \begin{array}{l} \text{heads}.b \twoheadrightarrow a(\text{win}.X), \\ \text{tails}.b \twoheadrightarrow a(\text{lose}.\text{end}) \end{array} \right), \\ \text{tails}.c \twoheadrightarrow b \left( \begin{array}{l} \text{heads}.b \twoheadrightarrow a(\text{lose}.\text{end}), \\ \text{tails}.b \twoheadrightarrow a(\text{win}.X) \end{array} \right) \end{array} \right)$$

# Implementing MPST as communicating processes

- Participant implementations consist of *protocol actions*, denoting the input and output of values and labels. E.g., $a[\text{heads}]$ and $b(\text{heads})$.

- A *communication function* derived from the global type allows processes to communicate. E.g., $a[\text{heads}] \mid b(\text{heads}) \mapsto ab\langle\text{heads}\rangle$.

$$G_{\text{coin}} := \text{rec}\, X\,.\, a \twoheadrightarrow b\langle\text{nat}\rangle\,.\, a \twoheadrightarrow b \left( \begin{array}{l} \text{heads}\,.\, c \twoheadrightarrow b \left( \begin{array}{l} \text{heads}\,.\, b \twoheadrightarrow a(\text{win}\,.\, X), \\ \text{tails}\,.\, b \twoheadrightarrow a(\text{lose}\,.\, \text{end}) \end{array} \right), \\ \text{tails}\,.\, c \twoheadrightarrow b \left( \begin{array}{l} \text{heads}\,.\, b \twoheadrightarrow a(\text{lose}\,.\, \text{end}), \\ \text{tails}\,.\, b \twoheadrightarrow a(\text{win}\,.\, X) \end{array} \right) \end{array} \right)$$

# Implementing MPST as communicating processes

- Participant implementations consist of *protocol actions*, denoting the input and output of values and labels. E.g., $a[\text{heads}]$ and $b(\text{heads})$.
- A *communication function* derived from the global type allows processes to communicate. E.g., $a[\text{heads}] \mid b(\text{heads}) \mapsto ab\langle\text{heads}\rangle$.
- This is not enough: how to distinguish consecutive protocol actions? E.g., $c[\text{heads}] \mid b(\text{heads}) \mapsto cb\langle\text{heads}\rangle$.

$$G_{\text{coin}} := \text{rec}\, X\,.\, a \twoheadrightarrow b\langle\text{nat}\rangle\,.\, a \twoheadrightarrow b \left( \begin{array}{l} \text{heads}\,.\, c \twoheadrightarrow b \begin{pmatrix} \text{heads}\,.\, b \twoheadrightarrow a(\text{win}\,.\, X), \\ \text{tails}\,.\, b \twoheadrightarrow a(\text{lose}\,.\, \text{end}) \end{pmatrix}, \\ \text{tails}\,.\, c \twoheadrightarrow b \begin{pmatrix} \text{heads}\,.\, b \twoheadrightarrow a(\text{lose}\,.\, \text{end}), \\ \text{tails}\,.\, b \twoheadrightarrow a(\text{win}\,.\, X) \end{pmatrix} \end{array} \right)$$

# Implementing MPST as communicating processes

- A *communication function* derived from the global type allows processes to communicate. E.g., $a[\text{heads}] \mid b(\text{heads}) \mapsto ab\langle\text{heads}\rangle$.
- This is not enough: how to distinguish consecutive protocol actions? E.g., $c[\text{heads}] \mid b(\text{heads}) \mapsto cb\langle\text{heads}\rangle$.
- Our solution: number the exchanges in the global type, annotate protocol actions accordingly. E.g.,
  $a_2[\text{heads}] \mid b_2(\text{heads}) \mapsto ab_2\langle\text{heads}\rangle$ and
  $c_3[\text{heads}] \mid b_3(\text{heads}) \mapsto cb_3\langle\text{heads}\rangle$.

$$G_{\text{coin}} := \text{rec}\, X.\, a \overset{1}{\twoheadrightarrow} b\langle\text{nat}\rangle.\, a \overset{2}{\twoheadrightarrow} b \left( \begin{array}{l} \text{heads}.\, c \overset{3}{\twoheadrightarrow} b \left( \begin{array}{l} \text{heads}.\, b \overset{4}{\twoheadrightarrow} a(\text{win}.X), \\ \text{tails}.\, b \overset{5}{\twoheadrightarrow} a(\text{lose}.\overset{6}{\text{end}}) \end{array} \right), \\ \text{tails}.\, c \overset{7}{\twoheadrightarrow} b \left( \begin{array}{l} \text{heads}.\, b \overset{8}{\twoheadrightarrow} a(\text{lose}.\overset{9}{\text{end}}), \\ \text{tails}.\, b \overset{10}{\twoheadrightarrow} a(\text{win}.X) \end{array} \right) \end{array} \right)$$

# Implementing MPST as communicating processes

- Example participant implementations:

$$P_a := a_1[42] \,.\, a_2[\text{heads}] \,.\, (a_4(\text{win}) \,.\, P_a + a_5(\text{lose}) \,.\, a_6\text{end})$$

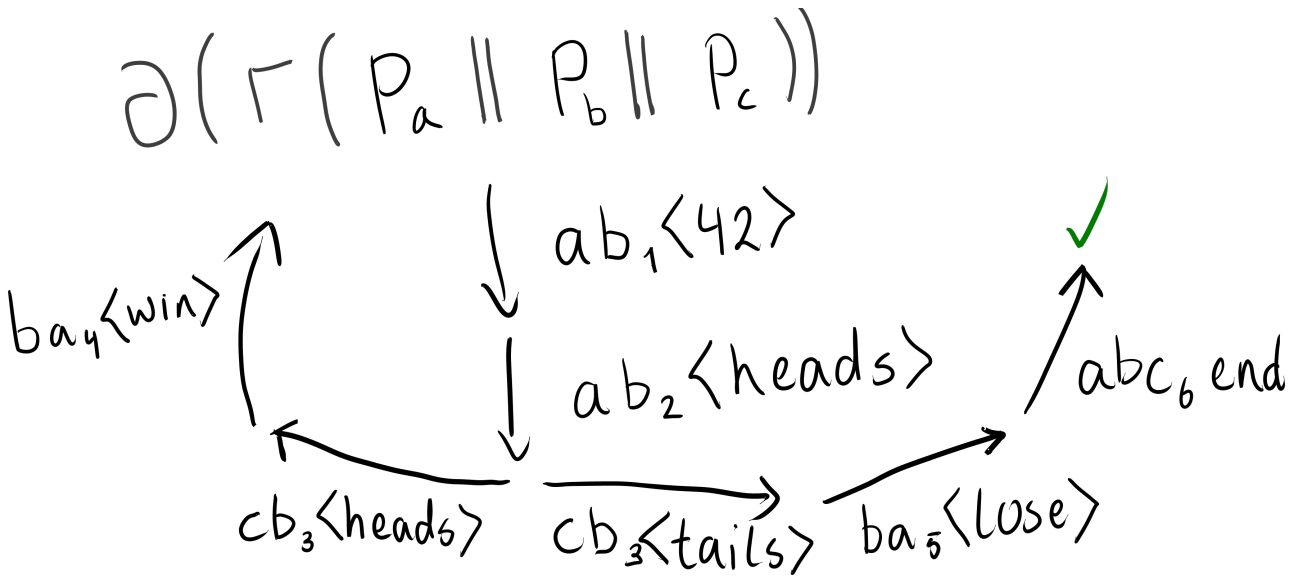
$$P_b := \sum_{x \in \mathbb{N}} b_1(x) \,.\, \left( \begin{array}{l} b_2(\text{heads}) \,.\, \left( \begin{array}{l} b_3(\text{heads}) \,.\, b_4[\text{win}] \,.\, P_b \\ + b_3(\text{tails}) \,.\, b_5[\text{lose}] \,.\, b_6\text{end} \end{array} \right) \\ + b_2(\text{tails}) \,.\, \left( \begin{array}{l} b_7(\text{heads}) \,.\, b_8[\text{lose}] \,.\, b_9\text{end} \\ + b_7(\text{tails}) \,.\, b_{10}[\text{win}] \,.\, P_b \end{array} \right) \end{array} \right)$$

$$P_c := c_3[\text{heads}] \,.\, P_c + c_3[\text{tails}] \,.\, c_6\text{end} \\ + c_7[\text{heads}] \,.\, c_9\text{end} + c_7[\text{tails}] \,.\, P_c$$

$$G_{\text{coin}} := \text{rec}\, X \,.\, a \overset{1}{\twoheadrightarrow} b\langle \text{nat} \rangle \,.\, a \overset{2}{\twoheadrightarrow} b \left( \begin{array}{l} \text{heads}.c \overset{3}{\twoheadrightarrow} b \left( \begin{array}{l} \text{heads}.b \overset{4}{\twoheadrightarrow} a(\text{win}.X), \\ \text{tails}.b \overset{5}{\twoheadrightarrow} a(\text{lose}.\overset{6}{\text{end}}) \end{array} \right), \\ \text{tails}.c \overset{7}{\twoheadrightarrow} b \left( \begin{array}{l} \text{heads}.b \overset{8}{\twoheadrightarrow} a(\text{lose}.\overset{9}{\text{end}}), \\ \text{tails}.b \overset{10}{\twoheadrightarrow} a(\text{win}.X) \end{array} \right) \end{array} \right)$$

# Implementing MPST as communicating processes

- Labeled transition system of global implementation:




$$G_{\text{coin}} := \operatorname{rec} X.\, a \overset{1}{\twoheadrightarrow} b\langle \text{nat} \rangle.\, a \overset{2}{\twoheadrightarrow} b \left( \begin{array}{l} \text{heads}.c \overset{3}{\twoheadrightarrow} b \left( \begin{array}{l} \text{heads}.b \overset{4}{\twoheadrightarrow} a(\text{win}.X), \\ \text{tails}.b \overset{5}{\twoheadrightarrow} a(\text{lose}.\overset{6}{\text{end}}) \end{array} \right), \\ \text{tails}.c \overset{7}{\twoheadrightarrow} b \left( \begin{array}{l} \text{heads}.b \overset{8}{\twoheadrightarrow} a(\text{lose}.\overset{9}{\text{end}}), \\ \text{tails}.b \overset{10}{\twoheadrightarrow} a(\text{win}.X) \end{array} \right) \end{array} \right)$$

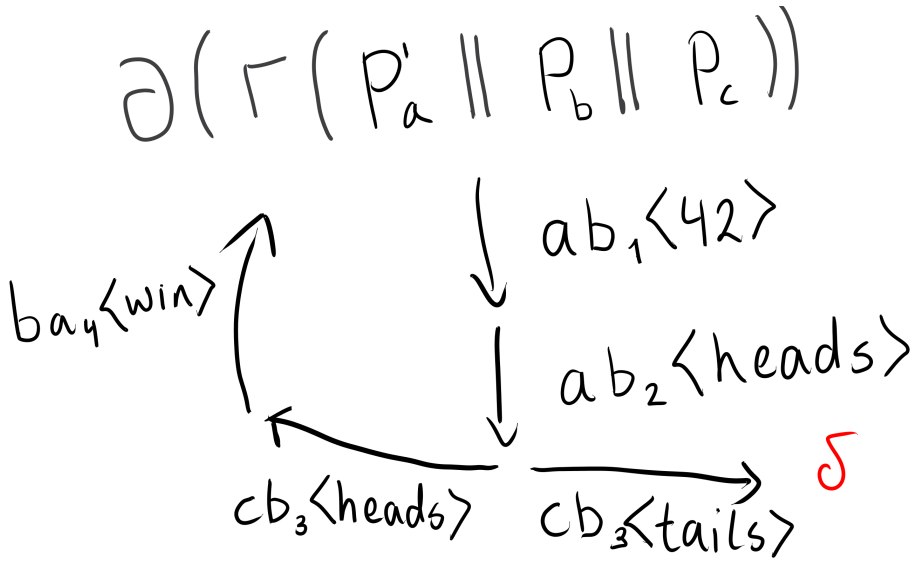
# Implementing MPST as communicating processes

- Alternative implementation of $a$: $P'_a := a_1[42] \,.\, a_2[\text{heads}] \,.\, a_4(win) \,.\, P'_a$.

$$G_{\text{coin}} := \text{rec}\, X \,.\, a \overset{1}{\twoheadrightarrow} b\langle \text{nat} \rangle \,.\, a \overset{2}{\twoheadrightarrow} b \left( \begin{array}{l} \text{heads}.c \overset{3}{\twoheadrightarrow} b \left( \begin{array}{l} \text{heads}.b \overset{4}{\twoheadrightarrow} a(\text{win}.X), \\ \text{tails}.b \overset{5}{\twoheadrightarrow} a(\text{lose}.\text{end}) \end{array} \right), \\ \text{tails}.c \overset{7}{\twoheadrightarrow} b \left( \begin{array}{l} \text{heads}.b \overset{8}{\twoheadrightarrow} a(\text{lose}.\text{end}), \\ \text{tails}.b \overset{10}{\twoheadrightarrow} a(\text{win}.X) \end{array} \right) \end{array} \right)$$

# Implementing MPST as communicating processes

- Alternative implementation of $a$: $P'_a := a_1[42] \,.\, a_2[\text{heads}] \,.\, a_4(win) \,.\, P'_a$.
- Labeled transition system:




$$G_{\text{coin}} := \text{rec}\, X \,.\, a \overset{1}{\twoheadrightarrow} b\langle \text{nat} \rangle \,.\, a \overset{2}{\twoheadrightarrow} b \left( \begin{array}{l} \text{heads} \,.\, c \overset{3}{\twoheadrightarrow} b \left( \begin{array}{l} \text{heads} \,.\, b \overset{4}{\twoheadrightarrow} a(\text{win} \,.\, X), \\ \text{tails} \,.\, b \overset{5}{\twoheadrightarrow} a(\text{lose} \,.\, \overset{6}{\text{end}}) \end{array} \right), \\ \text{tails} \,.\, c \overset{7}{\twoheadrightarrow} b \left( \begin{array}{l} \text{heads} \,.\, b \overset{8}{\twoheadrightarrow} a(\text{lose} \,.\, \overset{9}{\text{end}}), \\ \text{tails} \,.\, b \overset{10}{\twoheadrightarrow} a(\text{win} \,.\, X) \end{array} \right) \end{array} \right)$$

# Current work

- Model checking of process implementations.

# Current work

- Model checking of process implementations.
  - Well-formedness conditions for global types,

# Current work

- Model checking of process implementations.
  - Well-formedness conditions for global types,
  - Formulas for checking *protocol conformance and safety* of implementations:

# Current work

- Model checking of process implementations.
  - Well-formedness conditions for global types,
  - Formulas for checking *protocol conformance and safety* of implementations:
    *Local formulas* for checking the correctness of individual participant implementations,
    *Global formulas* for checking the correctness of global implementations.

# Current work

- For an exchange from $s$ to $r$ (type $T$ or labels $l$), the **global formula** verifies that:

# Current work

- For an exchange from $s$ to $r$ (type $T$ or labels $I$), the **global formula** verifies that:
  - There is a communication from $s$ to $r$ carrying any value of type $T$ or label in $I$, annotated with the appropriate exchange number,

# Current work

- For an exchange from $s$ to $r$ (type $T$ or labels $I$), the **global formula** verifies that:
  - There is a communication from $s$ to $r$ carrying any value of type $T$ or label in $I$, annotated with the appropriate exchange number,
  - After any such communication, the formula for the continuation holds,

# Current work

- For an exchange from $s$ to $r$ (type $T$ or labels $I$), the **global formula** verifies that:
  - There is a communication from $s$ to $r$ carrying any value of type $T$ or label in $I$, annotated with the appropriate exchange number,
  - After any such communication, the formula for the continuation holds,
  - There are no other communications possible.

# Current work

- For an exchange from $s$ to $r$ (type $T$ or labels $I$), the **global formula** verifies that:
  - There is a communication from $s$ to $r$ carrying any value of type $T$ or label in $I$, annotated with the appropriate exchange number,
  - After any such communication, the formula for the continuation holds,
  - There are no other communications possible.
- Possible issue: independent exchanges, e.g., $a \twoheadrightarrow b\langle T \rangle \,.\, c \twoheadrightarrow d\langle T' \rangle \ldots$.

# Current work

- For an exchange from $s$ to $r$ (type $T$ or labels $I$), the **global formula** verifies that:
  - There is a communication from $s$ to $r$ carrying any value of type $T$ or label in $I$, annotated with the appropriate exchange number,
  - After any such communication, the formula for the continuation holds,
  - There are no other communications possible.
- Possible issue: independent exchanges, e.g., $a \twoheadrightarrow b\langle T \rangle \,.\, c \twoheadrightarrow d\langle T' \rangle \ldots$.
- Solution (for now): rule out independent exchanges with well-formedness condition.

# Current work

- For an exchange from $s$ to $r$ (type $T$ or labels $l$), the **local formula** verifies that:

# Current work

- For an exchange from $s$ to $r$ (type $T$ or labels $I$), the **local formula** verifies that:
  - For $s$, there is an output carrying any value of type $T$ or label in $I$, annotated with the appropriate exchange number,

# Current work

- For an exchange from $s$ to $r$ (type $T$ or labels $I$), the **local formula** verifies that:
  - For $s$, there is an output carrying any value of type $T$ or label in $I$, annotated with the appropriate exchange number,
  - For $r$, there is an input for every value of type $T$ or label in $I$, annotated with the appropriate exchange number,

# Current work

- For an exchange from $s$ to $r$ (type $T$ or labels $I$), the **local formula** verifies that:
  - For $s$, there is an output carrying any value of type $T$ or label in $I$, annotated with the appropriate exchange number,
  - For $r$, there is an input for every value of type $T$ or label in $I$, annotated with the appropriate exchange number,
  - For $s$ and $r$, after any such protocol action, the formula for the continuation holds,

# Current work

- For an exchange from $s$ to $r$ (type $T$ or labels $I$), the **local formula** verifies that:
  - For $s$, there is an output carrying any value of type $T$ or label in $I$, annotated with the appropriate exchange number,
  - For $r$, there is an input for every value of type $T$ or label in $I$, annotated with the appropriate exchange number,
  - For $s$ and $r$, after any such protocol action, the formula for the continuation holds,
  - For $s$ and $r$, there are no other protocol actions possible,

# Current work

- For an exchange from $s$ to $r$ (type $T$ or labels $I$), the **local formula** verifies that:
  - For $s$, there is an output carrying any value of type $T$ or label in $I$, annotated with the appropriate exchange number,
  - For $r$, there is an input for every value of type $T$ or label in $I$, annotated with the appropriate exchange number,
  - For $s$ and $r$, after any such protocol action, the formula for the continuation holds,
  - For $s$ and $r$, there are no other protocol actions possible,
  - For participants other than $s$ and $r$, the formula of the continuation holds (conjunction of the continuations for labeled exchange).

# Current work

- For an exchange from $s$ to $r$ (type $T$ or labels $I$), the **local formula** verifies that:
  - For $s$, there is an output carrying any value of type $T$ or label in $I$, annotated with the appropriate exchange number,
  - For $r$, there is an input for every value of type $T$ or label in $I$, annotated with the appropriate exchange number,
  - For $s$ and $r$, after any such protocol action, the formula for the continuation holds,
  - For $s$ and $r$, there are no other protocol actions possible,
  - For participants other than $s$ and $r$, the formula of the continuation holds (conjunction of the continuations for labeled exchange).
- Possible issue: unawareness of branch picked, e.g., $a \twoheadrightarrow b(\ell_1 \,.\, b \twoheadrightarrow c\langle T\rangle \ldots, \ell_2 \,.\, b \twoheadrightarrow c\langle T'\rangle \ldots)$.

# Current work

- For an exchange from $s$ to $r$ (type $T$ or labels $I$), the **local formula** verifies that:
  - For $s$, there is an output carrying any value of type $T$ or label in $I$, annotated with the appropriate exchange number,
  - For $r$, there is an input for every value of type $T$ or label in $I$, annotated with the appropriate exchange number,
  - For $s$ and $r$, after any such protocol action, the formula for the continuation holds,
  - For $s$ and $r$, there are no other protocol actions possible,
  - For participants other than $s$ and $r$, the formula of the continuation holds (conjunction of the continuations for labeled exchange).
- Possible issue: unawareness of branch picked, e.g., $a \twoheadrightarrow b(\ell_1 \,.\, b \twoheadrightarrow c\langle T\rangle \ldots, \ell_2 \,.\, b \twoheadrightarrow c\langle T'\rangle \ldots)$.
- Solution: work in progress.

# Conclusion

- Done: approach to implementing global types as communicating processes in mCRL2.

# Conclusion

- Done: approach to implementing global types as communicating processes in mCRL2.
- Current work: formulas for model checking, and well-formedness conditions.

# Conclusion

- Done: approach to implementing global types as communicating processes in mCRL2.
- Current work: formulas for model checking, and well-formedness conditions.
- Pending work: develop compiler from global type specifications to mCRL2 projects.

# Conclusion

- Done: approach to implementing global types as communicating processes in mCRL2.
- Current work: formulas for model checking, and well-formedness conditions.
- Pending work: develop compiler from global type specifications to mCRL2 projects.
- Future work: support independent exchanges, and embed data constraints in global type specifications.

# Conclusion

- Done: approach to implementing global types as communicating processes in mCRL2.
- Current work: formulas for model checking, and well-formedness conditions.
- Pending work: develop compiler from global type specifications to mCRL2 projects.
- Future work: support independent exchanges, and embed data constraints in global type specifications.
- Questions or comments? Let's discuss now, or send us an email b.van.den.heuvel@rug.nl and j.a.perez@rug.nl.