



Towards Typechecking for Model Transformations by Monadic 2nd-Order Logic

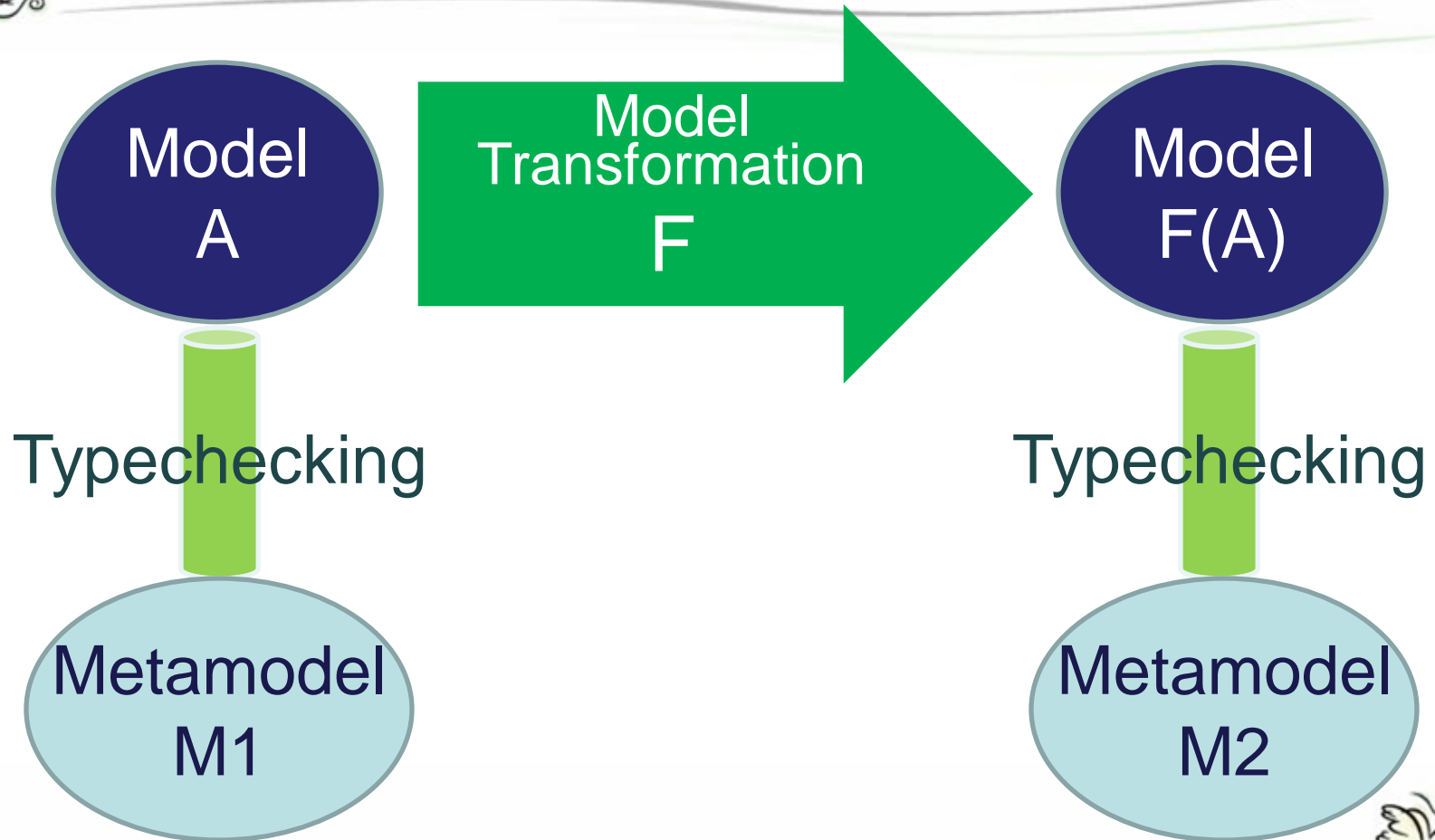
Kazuhiro Inaba (稲葉 一浩)
@ NII, BiG Team

Nov 16, 2009

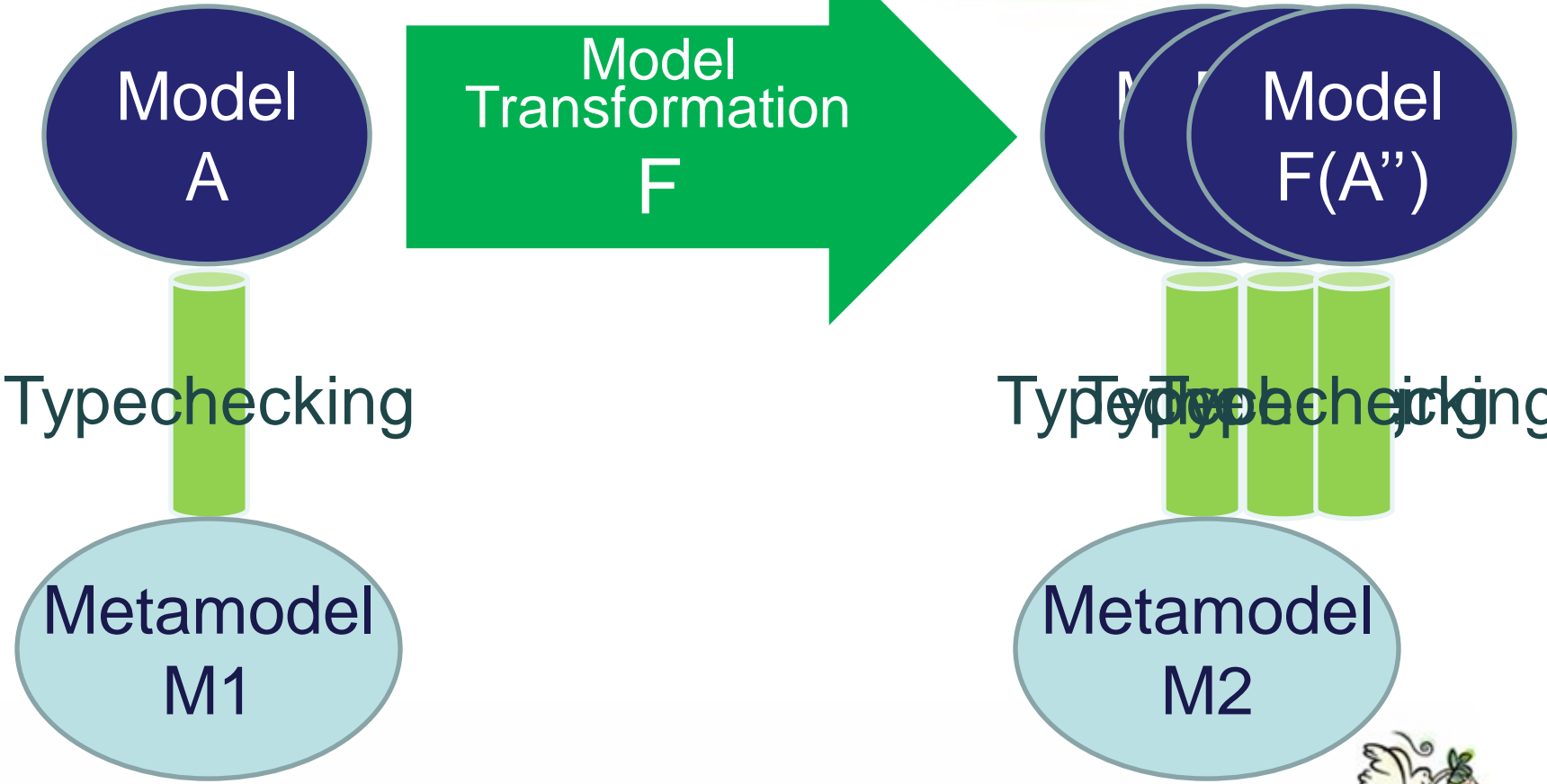
Changsha

3rd Bi-Trans in ABC

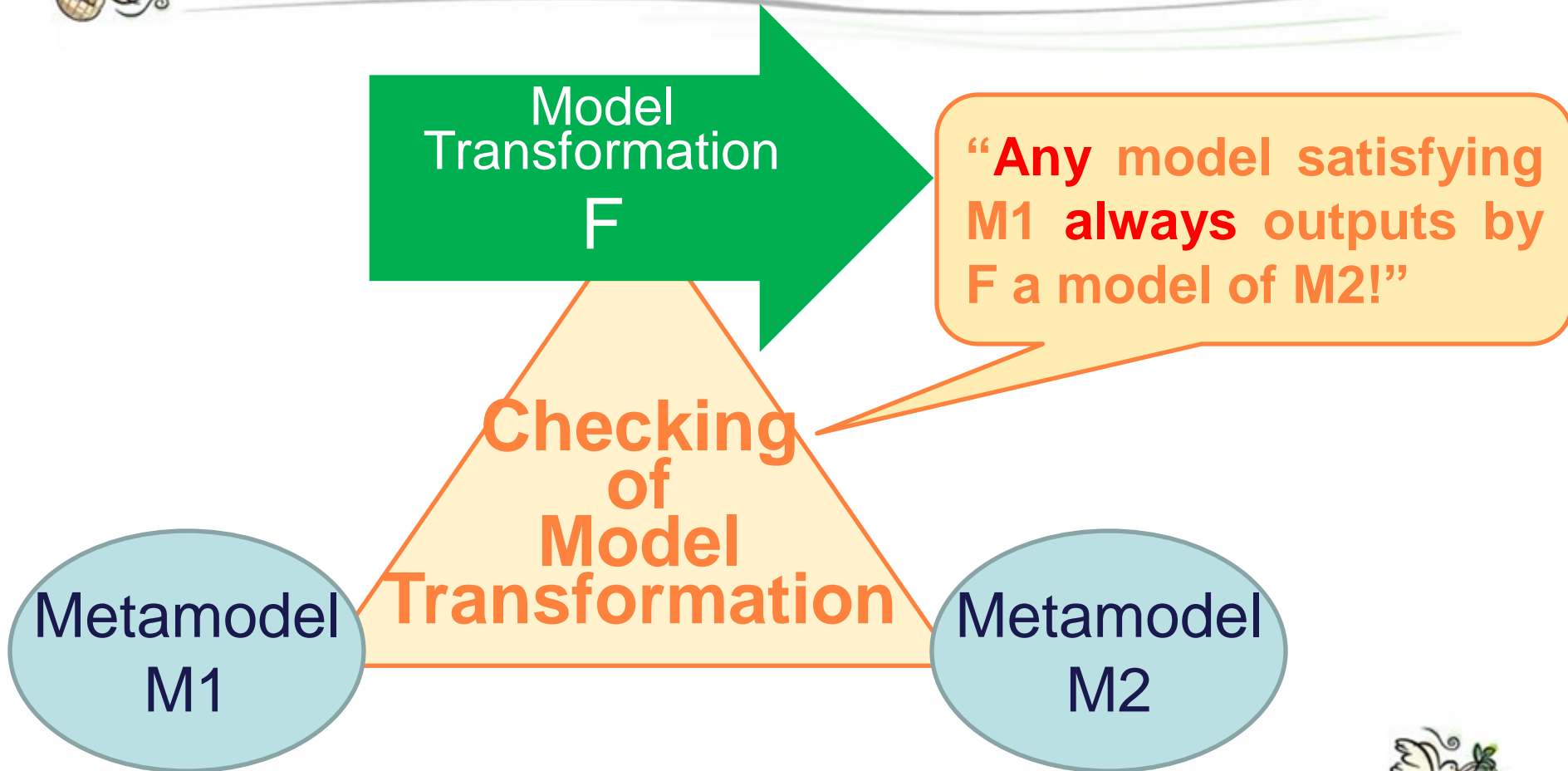
Checking Models Every Time



Checking Models Every Time



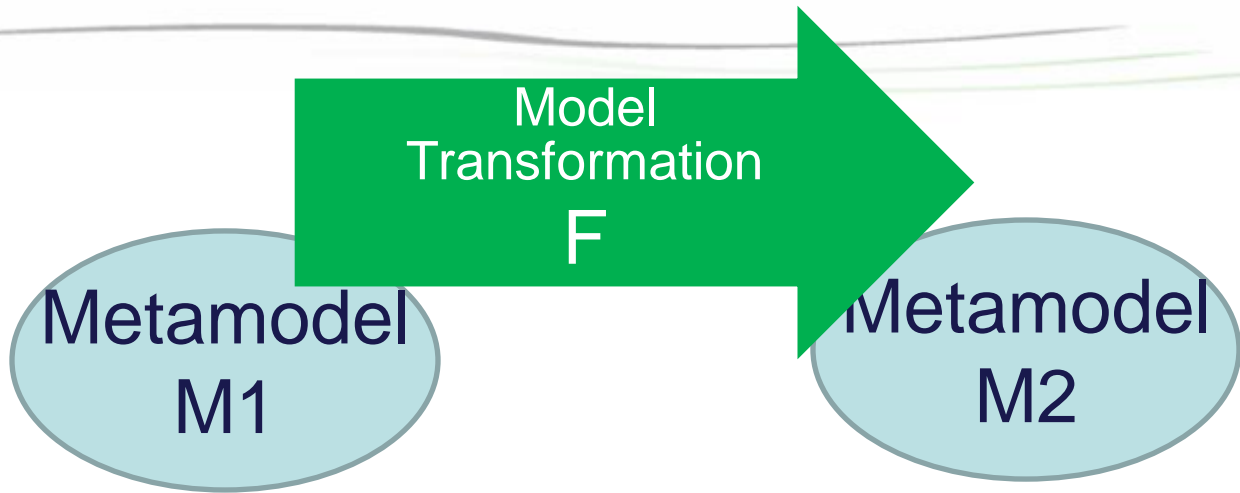
Checking Transformation Only Once!



Our Approach: Use Logic



Covert to
Logic
Formula



& “Input and Output are related by F”
“input satisfies M1” “output satisfies M2”

The text is contained within a grey rounded rectangle. A double arrow points from the right side of the text to the right.

Solver

Valid! (True for any model!)





Agenda

- 🌱 From Model Transformation to Logic
 - 🌱 MSO Logic, Graphs, Schemas and UnCAL
- 🌱 Validation of MSO Logic Formula
- 🌱 Conclusion and Future Work
 - 🌱 Application to Bidirectional Transformation





Monadic 2nd-Order Logic (MSO)

🍌 MSO is a

🍌 Usual 1st order predicate logic

🍌 Boolean ops and quantifiers: \neg , \wedge , \vee , \forall , \exists

🍌 ...extended with “set-quantifications”: \forall^{set} \exists^{set}

🍌 E.g.,, $\text{connected}(x,y) :=$

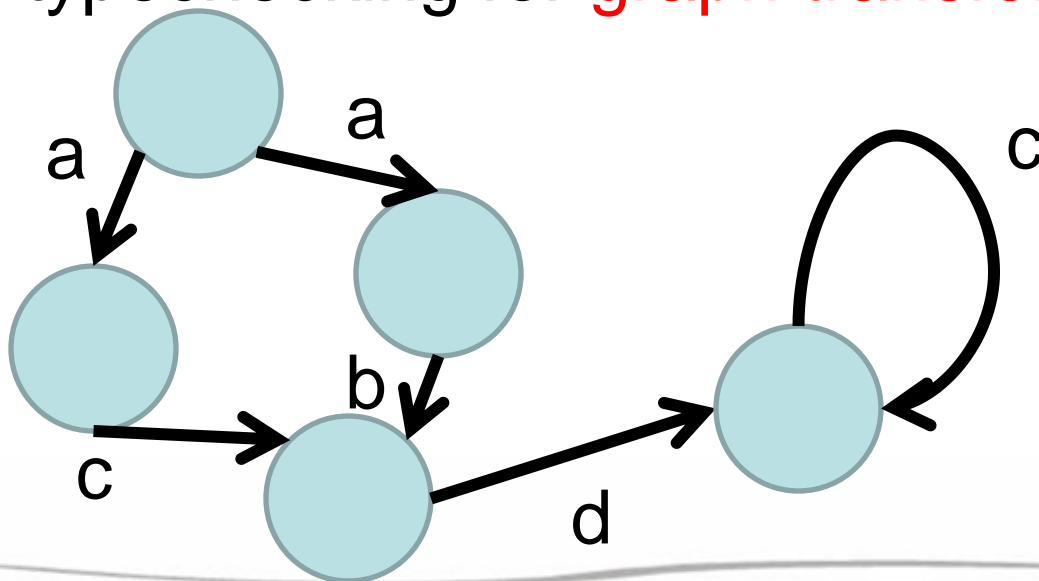
$$\begin{aligned} & \exists^{\text{set}} P. (x \in P \ \& \ y \in P \ \& \\ & \quad \forall u,v.(u \in P \ \& \ \text{edge}(u,v) \\ & \quad \Rightarrow v \in P) \ \& \ \dots) \end{aligned}$$





Graph (Model)

- 🍌 We regard models as edge-labeled graphs
 - ➔ i.e., we consider the problem of typechecking for **graph transformation**





Schema (Metamodel)

- Subset of KM3 [Jouault&Bezivin '06]

```
class Customer {  
    reference email [1-*] : String;  
    reference order [0-*] : Order;  
}  
class Order {  
    reference no [1-*] : Int;  
    reference order_of [1-*] : Customer;  
}
```

Only
[0-*] and
[1-*] is
allowed





Converting Schema to MSO

```
class Customer {
  reference email [1-*] : String;
  reference order [0-*] : Order; }
```

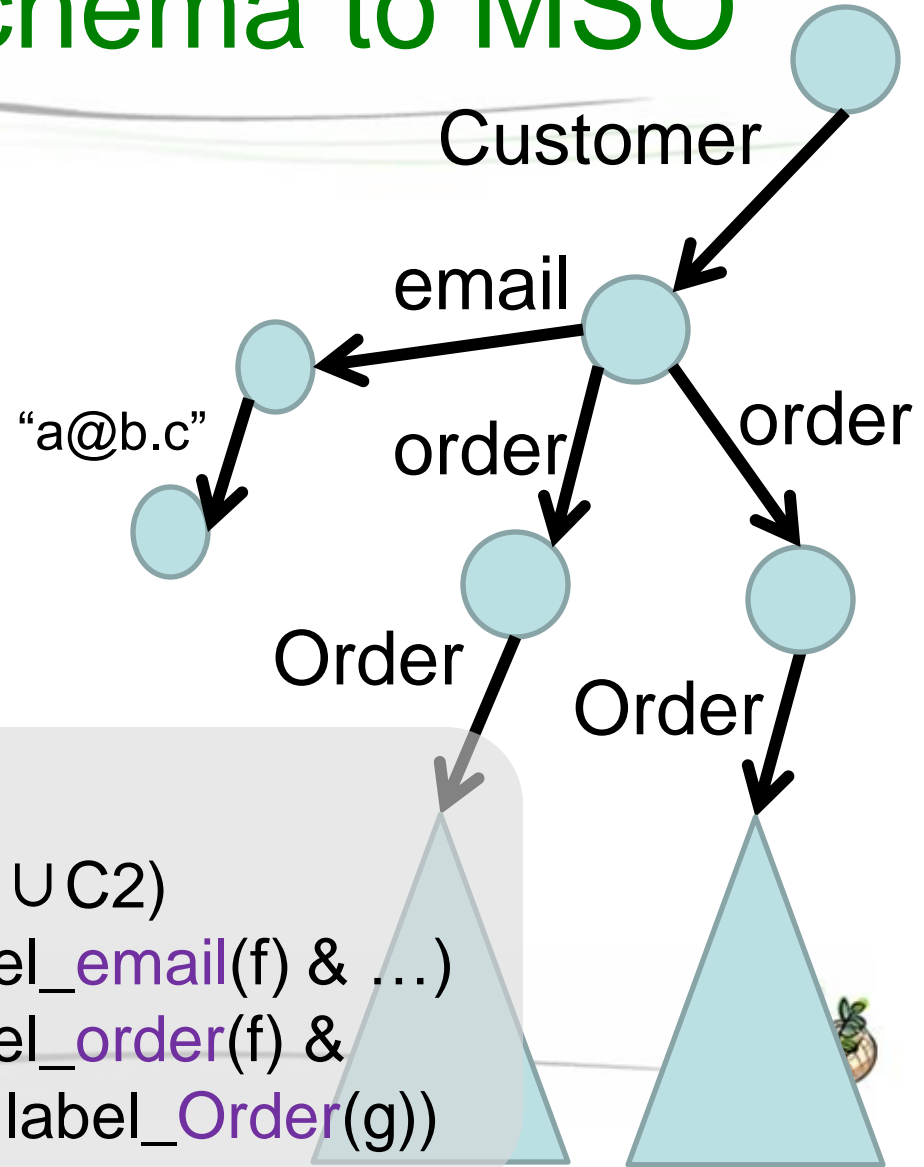


$$\forall e: \text{label_Customer}(e) \Rightarrow$$

$$\exists^{\text{set}} C1, C2. \text{outgoing}(e, C1 \cup C2)$$

$$\& |C1| \geq 1 \& \forall f \in C1. (\text{label_email}(f) \& \dots)$$

$$\& |C2| \geq 0 \& \forall f \in C2. (\text{label_order}(f) \&$$

$$\exists g. \text{outgoing}(f, \{g\}) \& \text{label_Order}(g))$$




Transformation Language

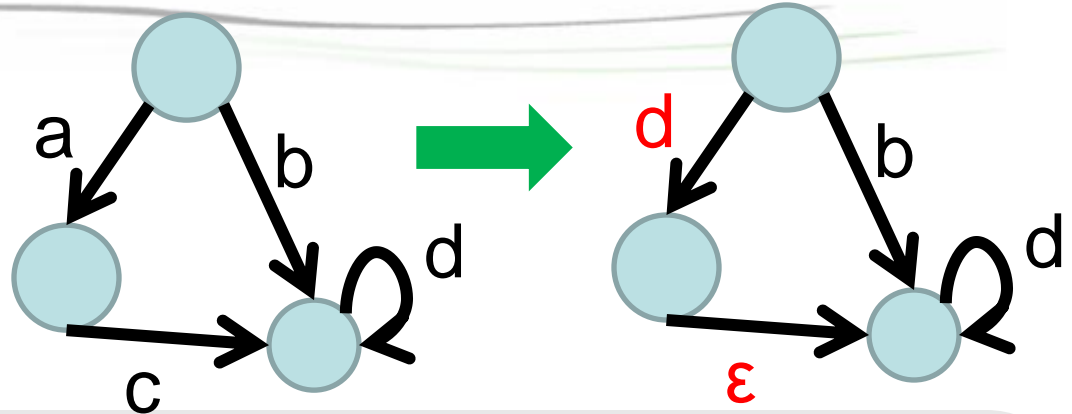
- 🌱 UnCAL [Buneman&Fernandez&Suiciu '00]
 - 🌱 Internal Graph Algebra of GRoundTram
 - 🌱 Based on “Structural Recursion” on graphs

```
rec( $\lambda$ ($L, _). // for each edge
  if $L = a then {d: &} // if label=a change to d
  else if $L = c then & // if label=c, delete
  else {$L: &} // otherwise, keep unchanged
)
```



Converting UnCAL to MSO

$\text{rec}(\lambda(\$L, _).$
if $\$L = a$ then $\{d: \&\}$
else if $\$L = c$ then $\&$
else $\{\$L: \&\}$)



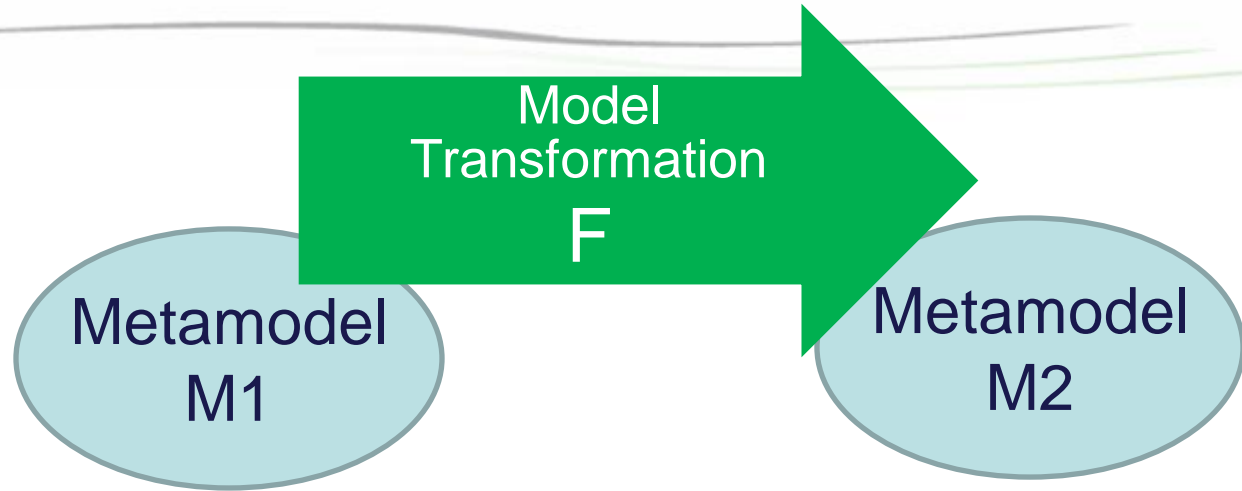
$\text{outgraph_label_d}(e)$

$\Leftrightarrow \text{ingraph_label_a}(e) \vee \text{ingraph_label_d}(e)$
 $\wedge \text{outgraph_label_a}(e) \Leftrightarrow \text{false}$
 $\wedge \text{outgraph_label_b}(e) \Leftrightarrow \text{ingraph_label_b}(e)$
 $\wedge \text{outgraph_label_c}(e) \Leftrightarrow \text{false}$
 $\wedge \text{outgraph_label_}\epsilon(e) \Leftrightarrow \text{ingraph_label_c}(e)$

[Revisited] Our Approach



Covert to
Logic
Formula



& “Input and Output are related by F”
“input satisfies M1” “output satisfies M2”


Solver

Valid! (True for any graph!)





Bad News

 Theorem [Trakhtenbrot 50]:
Validity property is **undecidable** on
graphs, even for 1st-order logic.

Valid! (True for any graph!)



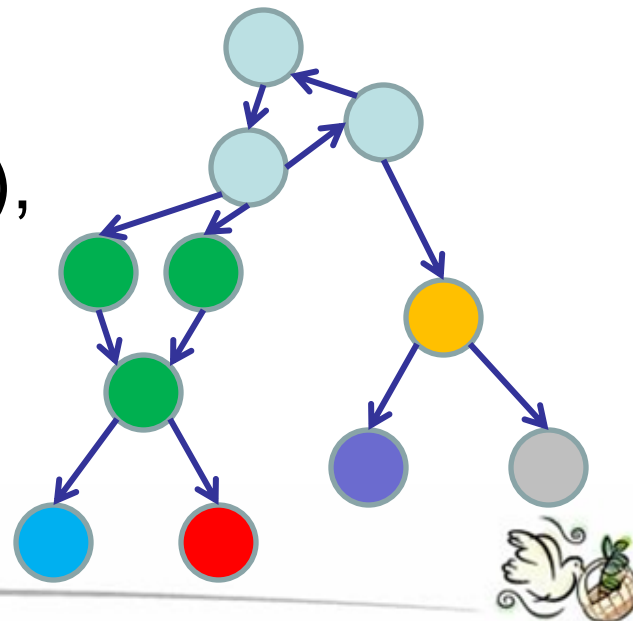


Not-so-bad News

🍌 MSO validness is **decidable on trees**.
[Thatcher&Wright68]

🍌 Also decidable on **tree-like** graphs (bounded tree-width), but it's *too* tree-like

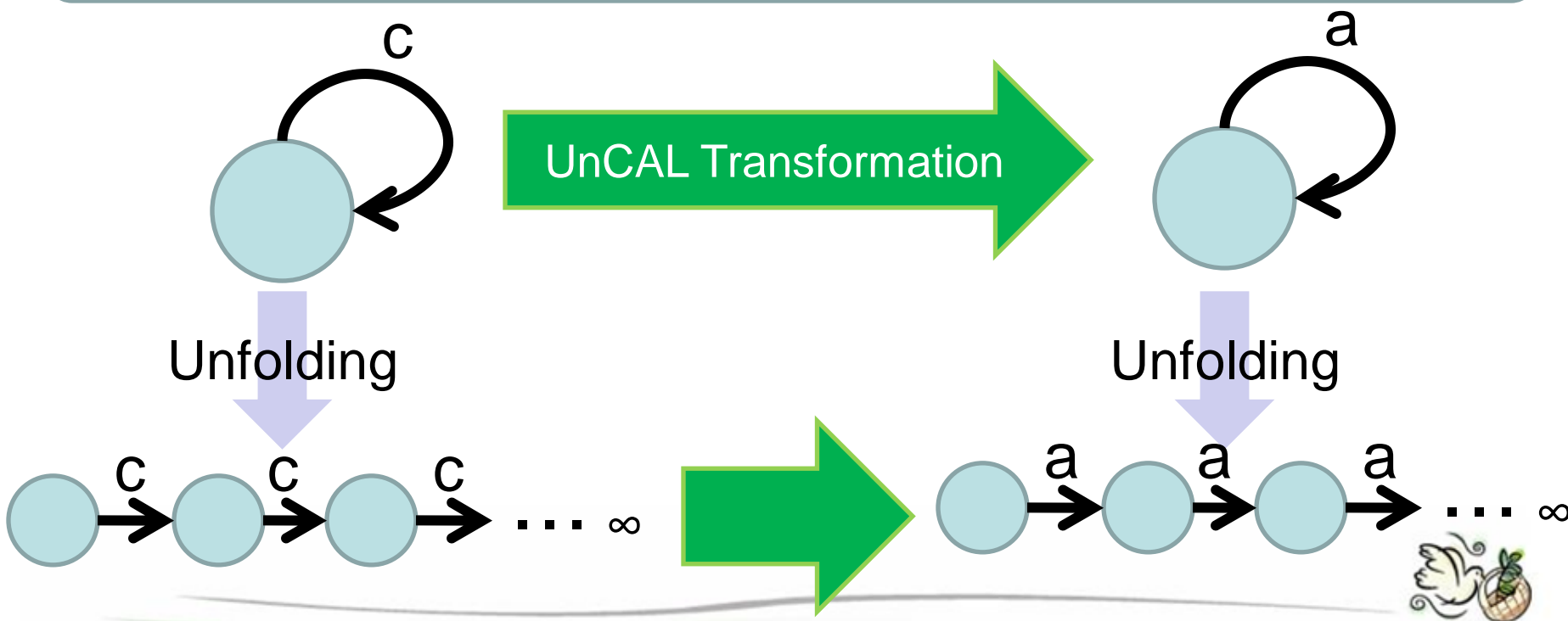
🍌 Models are general graphs!





Good News

🍌 UnCAL is **bisimulation-generic**. [BFS '00]





Our Approach: Infinite Trees

UnCAL and our schema do not distinguish a graph and its unfolded infinite tree

Our MSO formulas are valid (true on all graphs) iff true on all infinite trees

Decidable!

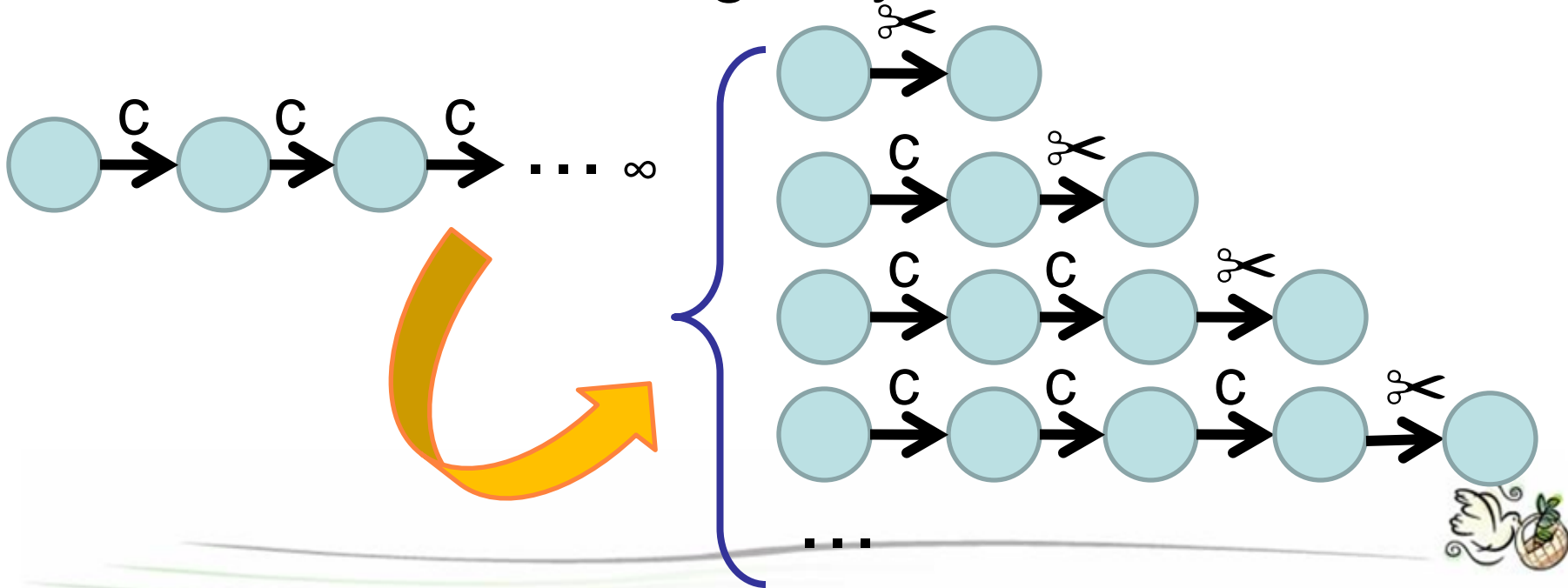


Our Approach:



Infinite Trees to Finite Trees

- 🍌 Bad: Deciding MSO on infinite trees is costly
- 🍌 Good: Considering only **Finite Cuts** suffices



Type Correctness



Validness of MSO Formula on Graphs

$$\Phi_{M1} \ \& \ \Phi_F \ \Rightarrow \ \Phi_{M2}$$

Validness of MSO Formula on Infinite Trees

$$\Phi_{M1} \ \& \ \Phi_F \ \Rightarrow \ \Phi_{M2}$$

Validness of MSO Formula on Finite Trees

$$\Phi'_{M1} \ \& \ \Phi'_F \ \Rightarrow \ \Phi'_{M2}$$

Checked by the Well-known Solver **MONA**



Demo

👉 (For showing the taste of the system)





Summary

- ☛ Typechecking is reduced to **MSO Validness**
- ☛ Not by restricting to tree-like graphs, but by exploiting the **Bisimulation-Genericity** of UnCAL





Ongoing Work

🍌 To Finish Implementation 😊

🍌 Performance Improvements

🍌 Supporting larger class of translation

🍌 Application to **Bidirectional Transformation**

🍌 “Updatability”: the following formula

$$\Phi'_{M1} \ \& \ \Phi'_F \ \& \ \Phi'_{M2}$$

defines the set of outputs having corresponding inputs





THANK YOU FOR LISTENING!

