

# Modal- $\mu$ Definable Graph Transduction

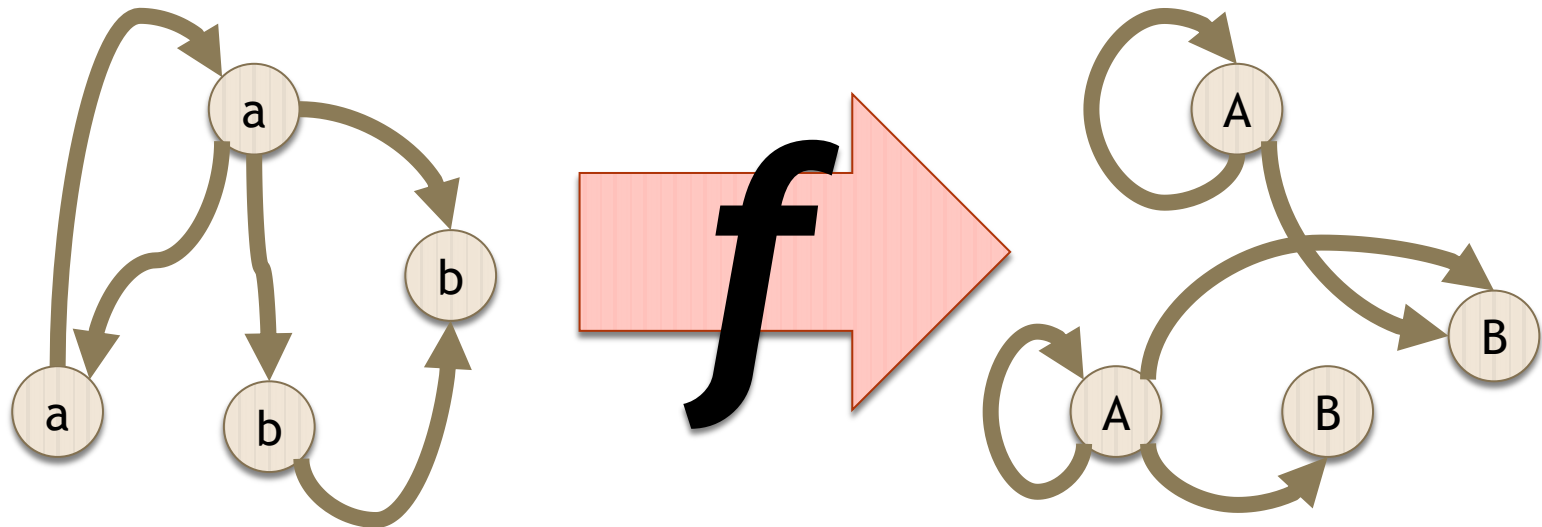
**Kazuhiro Inaba**

National Institute of Informatics, Japan

4<sup>th</sup> DIKU-IST Workshop, 2011

# What We Want to Do

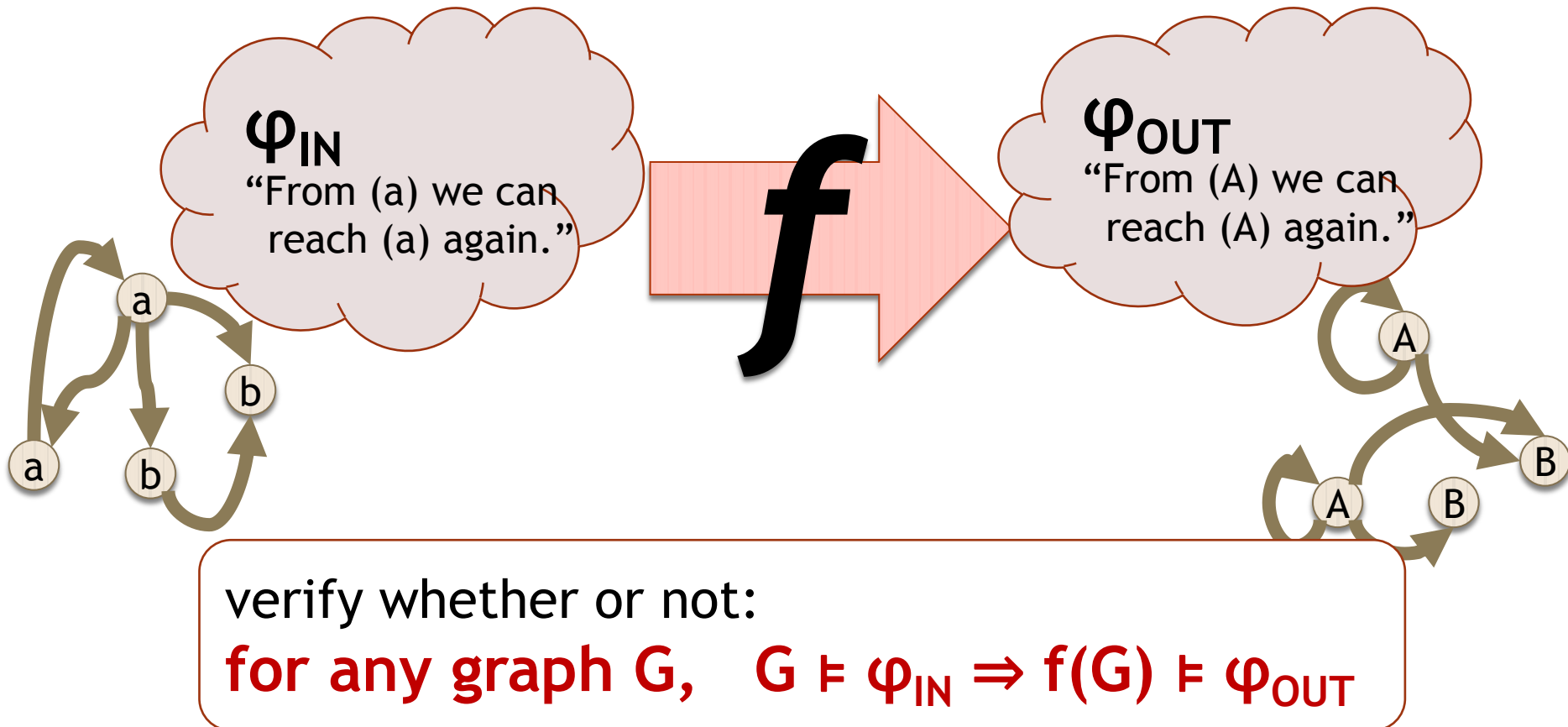
- Verification of **Graph-to-Graph Transformations**



- e.g., Queries on Graph-Structured Database  
or Transformations of XML with “id” links

# What We Want to Do

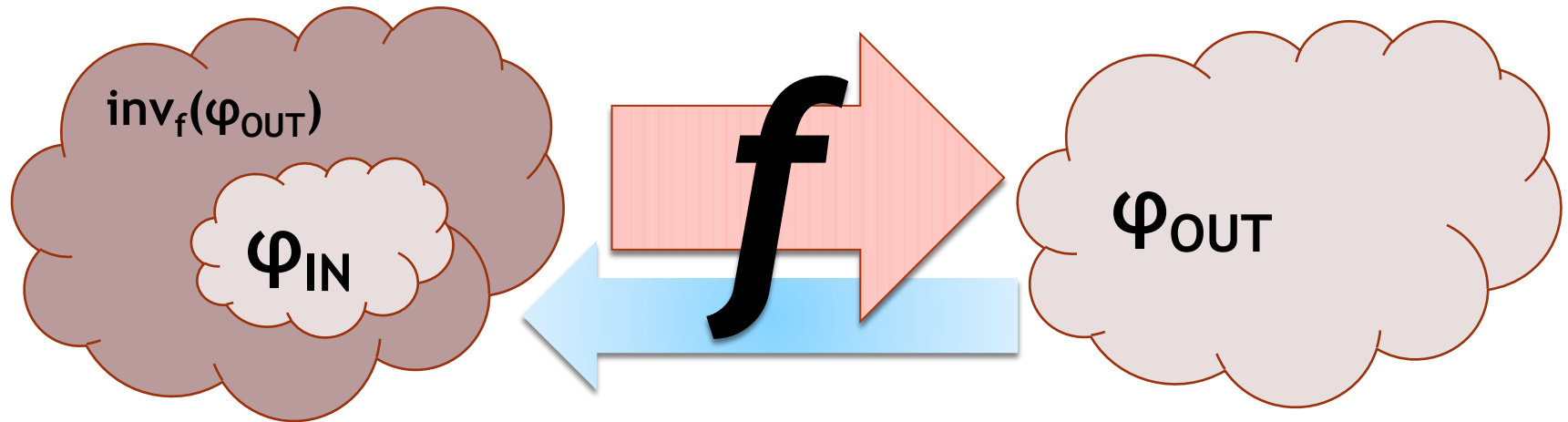
- Verification of Graph-to-Graph Transformations with respect to **input/output specifications**



# Verification by Pre-Image

(a.k.a. “weakest precondition” or “inverse type inference”)

Given  $f$  and  $\varphi_{\text{OUT}}$ , compute  $\text{inv}_f(\varphi_{\text{OUT}})$  such that:  
**for any graph  $G$ ,  $f(G) \models \varphi_{\text{OUT}}$  iff  $G \models \text{inv}_f(\varphi_{\text{OUT}})$**



Then “for any graph  $G$ ,  $G \models \varphi_{\text{IN}} \Rightarrow f(G) \models \varphi_{\text{OUT}}$ ”  
 iff “for any graph  $G$ ,  $G \models (\varphi_{\text{IN}} \rightarrow \text{inv}_f(\varphi_{\text{OUT}}))$ ”  
 i.e.,  $\varphi_{\text{IN}} \rightarrow \text{inv}_f(\varphi_{\text{OUT}})$  is *valid*

# To Be More Concrete...

- Which logic can we use for specifying  $\varphi_{\text{IN/OUT}}$  ?
  - Must be strong enough to express useful conditions.
  - Must be weak enough to have **decidable validity**.
- What kind of transformation  $f$  can be verified ?
  - We must be able to **compute the pre-image**.

# Our Approach

- Take **Modal- $\mu$  Calculus** as the specification logic
  - (At least for trees) capture all existing XML-Schemas
- Define a new model of graph transformation called **Modal- $\mu$  Definable Transduction**
  - Pre-image of modal- $\mu$  sentence can be fully automatically computed
  - Expressive enough to capture (unnested) structural recursion on graphs

# Related Work

- **MSO** (Monadic 2<sup>nd</sup>-Order Logic) Definable Transduction
  - Overall structure is more or less the same.
  - Ours is a proposal to use **Modal- $\mu$  instead of MSO**
- Hoare-Style Verification of Imperative Programs
  - Ours don't deal with pointers or destructive updates.
  - Rather, it is more suitable for **functional programs**
  - **Structural recursion** is handled without any annotations

```

fun f( {$l: $x} ) = {cap($l) : g($x)}
fun g( {_: $x} ) = f($x)
{  $\varphi_{IN}$  } f {  $\varphi_{OUT}$  }

```

```

{  $\varphi_{IN}$  }
p := root
while p != null do
  q := p.next
  p.next := p.next.next
  p := q
end
{  $\varphi_{OUT}$  }

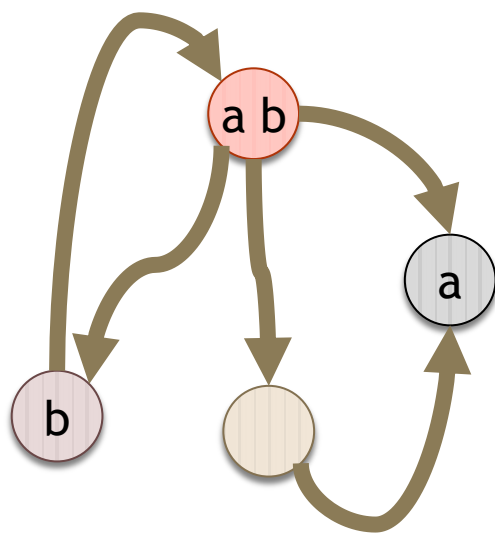
```

# Outline

- Two Kinds of Logics on Graphs
  - Predicate Logics
  - Modal Logics
  - **Why Modal- $\mu$  ?**
- Review: Predicate-Logic Based Approach
  - MSO-Definable Graph Transduction [Courcelle 94]
- **Our Work:**
  - **Modal- $\mu$  Definable Graph Transduction**
  - **Computation of Pre-Image**

# Graphs (in Today's Talk)

- $\Sigma$  : Finite Nonempty Alphabet
- $G = (V, E, \pi)$ 
  - $V$  Set of Nodes
  - $E \subseteq V \times V$  Set of Directed Edges
  - $\pi : V \rightarrow 2^\Sigma$  Labels on Nodes



=

$$\Sigma = \{a, b\} \quad \pi =$$

$$V = \left\{ \begin{array}{l} \text{a b} \\ \text{a} \\ \text{b} \\ \text{ } \end{array} \right\} \begin{array}{l} \rightarrow \{a, b\} \\ \rightarrow \{a\} \\ \rightarrow \{b\} \\ \rightarrow \{\} \end{array}$$

# Predicate Logics on Graphs

$\varphi ::=$

FO

| False |  $\neg \varphi$  |  $\varphi \vee \varphi$

|  $\sigma(x)$  (for  $\sigma \in \Sigma$ )      “node  $x$  is labeled  $\sigma$ ”

| edge( $x, y$ )      “an edge connects  $x$  to  $y$ ”

|  $\exists x. \varphi$       “there’s  $x$  that makes  $\varphi$  hold”

|  $\exists S. \varphi$       “there’s a set  $S$  that makes  $\varphi$  hold”

|  $x \in S$       “ $x$  is in  $S$ ”

MSO

*We can define True,  $\varphi \wedge \varphi$ ,  $\varphi \rightarrow \varphi$ ,  $\forall x. \varphi$ , and  $\forall S. \varphi$ .*

# Semantics

- For a graph  $G=(V,E,\pi)$  and an environment  $\Gamma : \text{Var} \rightarrow V$ 
  - $G, \Gamma \models \sigma(x)$  iff  $\sigma \in \pi(\Gamma(x))$   
“node  $x$  is labeled  $\sigma$ ”
  - $G, \Gamma \models \text{edge}(x, y)$  iff  $(\Gamma(x), \Gamma(y)) \in E$   
“an edge connects  $x$  to  $y$ ”
  - $G, \Gamma \models \exists x. \varphi$  iff there's  $v \in V$  s.t.  $G, \Gamma[x:v] \models \varphi$
  - ...

# Modal Logics on Graphs

$\psi ::=$

| False |  $\neg \varphi$  |  $\varphi \vee \varphi$

|  $\sigma$  (for  $\sigma \in \Sigma$ ) “current node is labeled  $\sigma$ ”

|  $\diamond \varphi$  “current node has an outgoing edge whose destination satisfies  $\varphi$ ”

|  $X$

|  $\mu X. \varphi$  “least fixpoint” (X must be in even # of  $\neg$ )

M

$M\mu$

*We Can Define:  $\square \varphi$  (dual of  $\diamond$ ) and  $\nu X. \varphi$  (GreatestFixPt)*

# Semantics

- For a graph  $G=(V,E,\pi)$ , an environment  $\Gamma : \text{Var} \rightarrow 2^V$ , and the current node  $v \in V$ 
  - $\mathbf{G, v, \Gamma \models \sigma}$  iff  $\sigma \in \pi(v)$   
“current node is labeled  $\sigma$ ”
  - $\mathbf{G, v, \Gamma \models \Diamond \varphi}$  iff there's  $w (v,w) \in E$  &  $G, w, \Gamma \models \varphi$   
“current node has an outgoing edge whose destination satisfies  $\varphi$ ”
  - $\mathbf{G, v, \Gamma \models \mu Y. \varphi}$  iff  $v \in \text{LFP}(F)$   
where  $F(A) = \{w \in V \mid G, w, \Gamma[Y:A] \models \varphi\}$

...

# Examples

- “From the node  $x$ , we can reach a  $\sigma$ -node”

$$\forall S. ( (x \in S \wedge \forall y. \forall z. (y \in S \wedge \text{edge}(y,z) \rightarrow z \in S)) \rightarrow \exists y. (y \in S \wedge \sigma(y)))$$

- “Confluence”

$$\forall y. \forall z. ( \text{edge}(x,y) \wedge \text{edge}(x,z) \rightarrow \exists w. (\text{edge}(y,w) \wedge \text{edge}(z,w)) )$$

- “From the current node, we can reach a  $\sigma$ -node”

$$\mu Y. (\sigma \vee \diamond Y)$$

- “Confluence”

(No way to express it in Modal- $\mu$ )

# MSO Definable (1-copying) Transduction [Courcelle 94]

A set of MSO formulas  $T =$

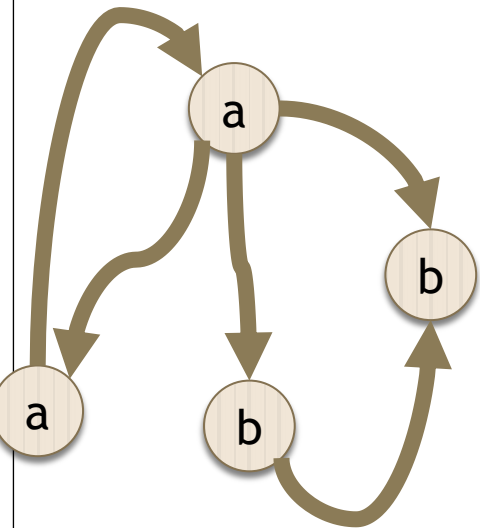
- $\sigma_{\text{OUT}}(\mathbf{x})$  for each  $\sigma \in \Sigma$
- $\text{edge}_{\text{OUT}}(\mathbf{x}, \mathbf{y})$

defines a transformation  $f_T$  converting

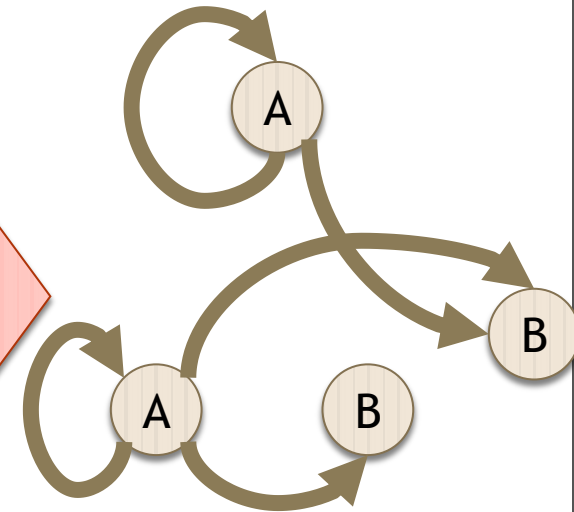
$G = (V, E, \pi)$  into  $G' = (V, E', \pi')$  where

- $\pi'(v) = \{ \sigma \mid G, \mathbf{x}:v \models \sigma_{\text{OUT}}(\mathbf{x}) \}$
- $E' = \{ (v, w) \mid G, \mathbf{x}:v, \mathbf{y}:w \models \text{edge}_{\text{OUT}}(\mathbf{x}, \mathbf{y}) \}$

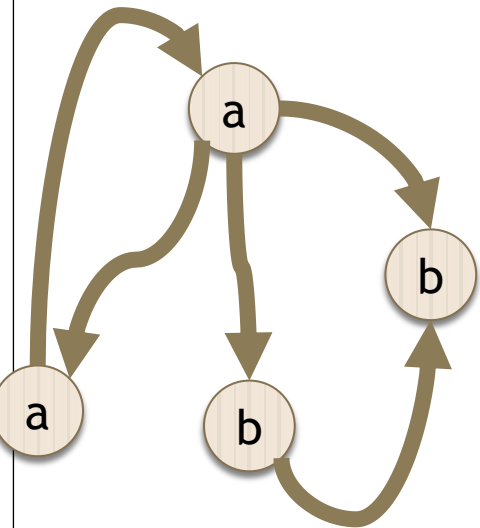
# Example ( $\Sigma = \{a, b, A, B\}$ )



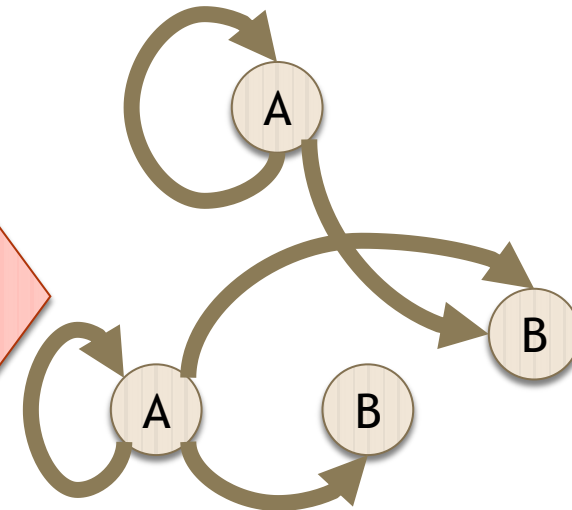
$\text{edge}_{\text{OUT}}(x, y) \equiv$   
 $\exists z. (\text{edge}(x, z) \wedge \text{edge}(z, y))$   
 $a_{\text{OUT}}(x) \equiv b_{\text{OUT}}(x) \equiv \text{False}$   
 $A_{\text{OUT}}(x) \equiv a(x)$   
 $B_{\text{OUT}}(x) \equiv b(x)$



# Pre-Image is Easily Obtained



$\text{edge}_{\text{OUT}}(x, y) \equiv$   
 $\exists z. (\text{edge}(x, z) \wedge \text{edge}(z, y))$   
 $a_{\text{OUT}}(x) \equiv b_{\text{OUT}}(x) \equiv \text{False}$   
 $A_{\text{OUT}}(x) \equiv a(x)$   
 $B_{\text{OUT}}(x) \equiv b(x)$

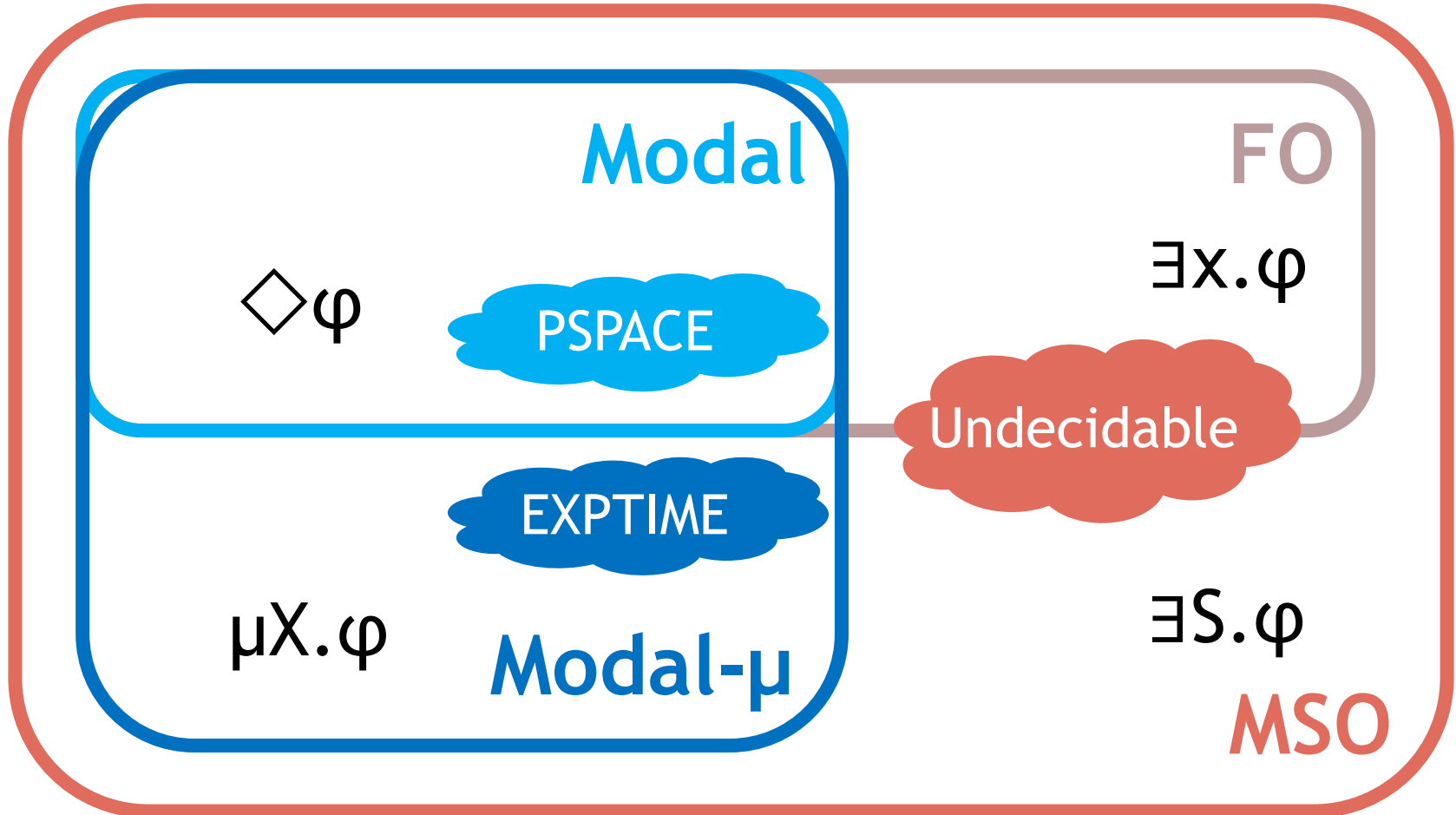


$\forall x. a(x) \rightarrow \exists y.$   
 $\exists z. (\text{edge}(x, z) \wedge \text{edge}(z, y))$   
 $\wedge a(y)$

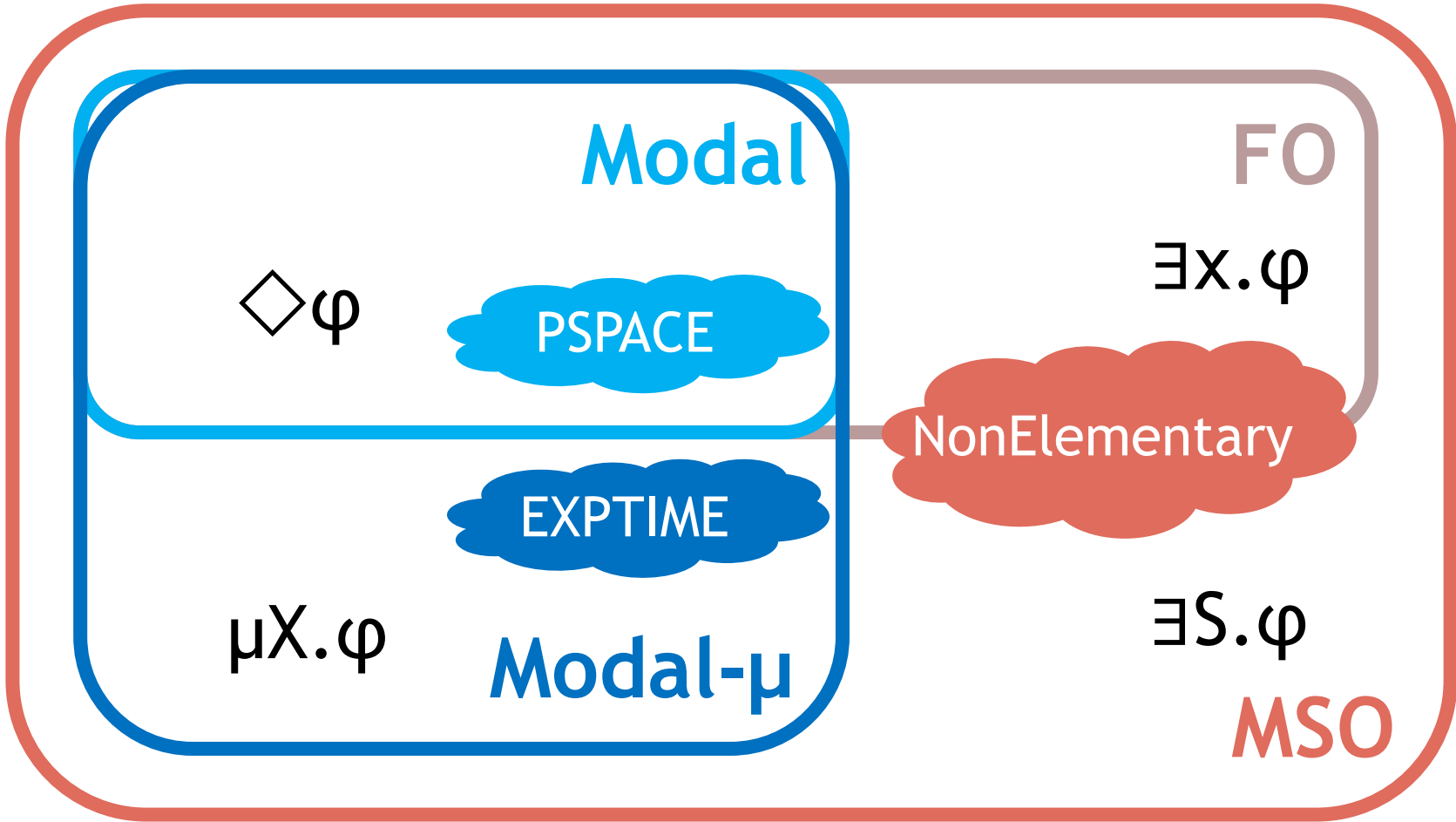
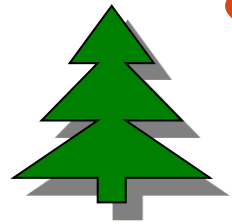
*Inline  
Expansion*

$\forall x. A(x) \rightarrow \exists y.$   
 $\text{edge}(x, y)$   
 $\wedge A(y)$

# Expressiveness & Complexity



# Expressiveness & Complexity (on “tree-like” graphs)



# Modal- $\mu$ and MSO

- Complexity of Validity Checking
  - **Modal- $\mu$  : EXPTIME-complete**
  - MSO : Undecidable (Even in Trees, HyperEXP)
- Expressive Power
  - Modal- $\mu$  = ***Bisimulation-Invariant Subset of MSO***  
[Janin & Walukiewicz 96]
  - “Bisimulation-Invariant”  $\approx$   
“Physical equality of pointers cannot be checked”
  - Not a severe restriction for purely functional programs!

# Modal- $\mu$ Definable (1-copying) Transduction

A set of Modal- $\mu$  formulas  $T =$

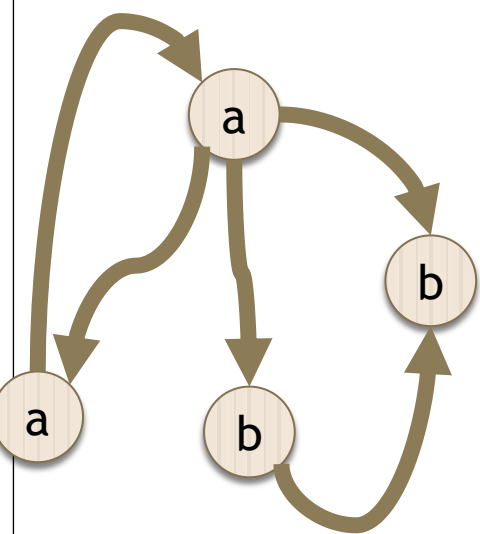
- $\sigma_{\text{OUT}}$  for each  $\sigma \in \Sigma$
- $\text{edge}_{\text{OUT}}$  an *existential* formula  $Fv=\{X\}$

defines a transformation  $f_T$  converting

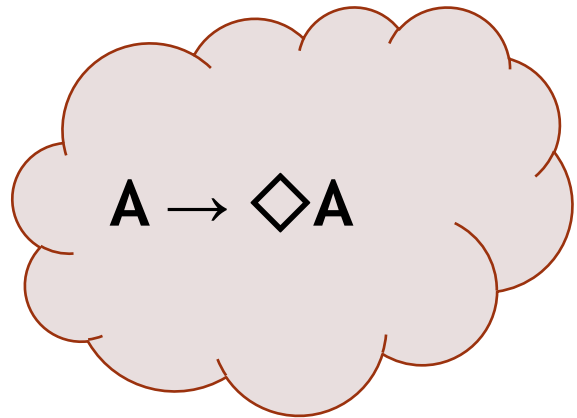
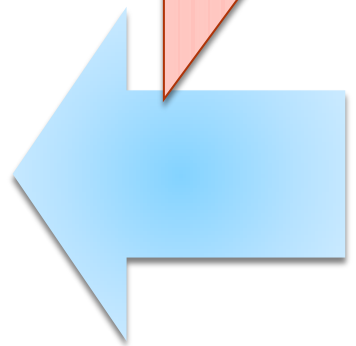
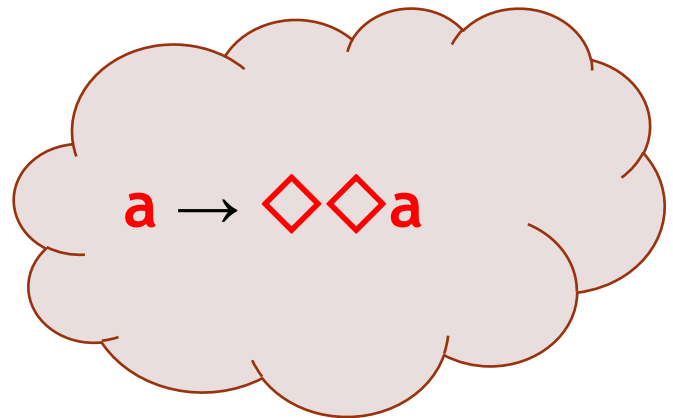
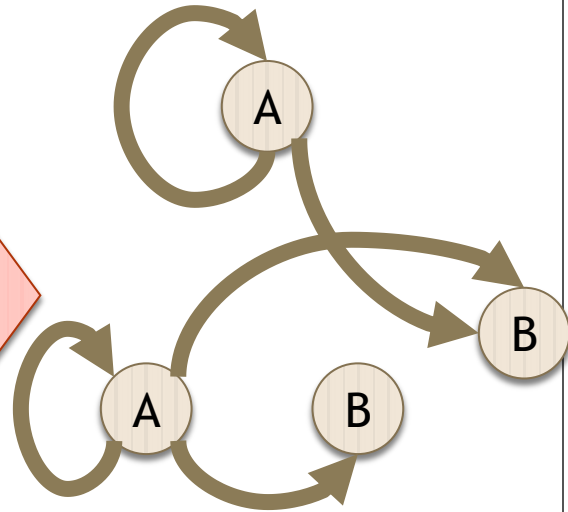
$G = (V, E, \pi)$  into  $G' = (V, E', \pi')$  where

- $\pi'(v) = \{ \sigma \mid G, v \models \sigma_{\text{OUT}} \}$
- $E' = \{ (v, w) \mid G, v, X:\{w\} \models \text{edge}_{\text{OUT}} \}$

# Example ( $\Sigma = \{a, b, A, B\}$ )



**edge**<sub>OUT</sub>  $\equiv \diamond\diamond X$   
**a**<sub>OUT</sub>  $\equiv$  **b**<sub>OUT</sub>  $\equiv$  False  
**A**<sub>OUT</sub>  $\equiv$  **a**  
**B**<sub>OUT</sub>  $\equiv$  **b**



# Existential Formula

- A formula  $e$  with one free variable  $X$  is *existential*, if

for all  $G=(V,E,\pi)$ ,  $v \in V$ ,  $P \subseteq V$   
 $G, v, X:P \models e$     iff     $\exists w \in P. G, v, X:\{w\} \models e$

- Examples:
  - “ $X \vee \text{True}$ ” is not existential (Consider  $P=\{\}$ ).
  - “ $\diamond X$ ” is existential.
  - “ $\square X$ ” is not (when  $v$  is a leaf node ...).
  - “ $\sigma$ ” is not, but “ $X \wedge \sigma$ ” is.

# Syntactic Condition

for all  $G=(V,E,\pi)$ ,  $v \in V$ ,  $P \subseteq V$   
 $G, v, X:P \models e$  iff  $\exists w \in P. G, v, X:\{w\} \models e$

- Theorem:  
 $e$  is existential if it is in the following syntax

$e ::= \text{False} \mid X \mid Y \mid e \vee e \mid \diamond e \mid \mu Y. e$   
 $\mid e \wedge \varphi$  where  $\varphi$  is any formula without free variables

(True,  $\neg$ ,  $\sigma$ ,  $\square$ , and GFP must be “guarded” by  $\_ \wedge \_$ )

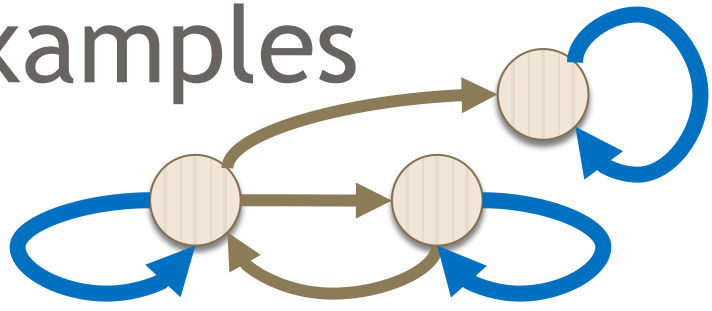


OPEN QUESTION: is this a necessary condition ?

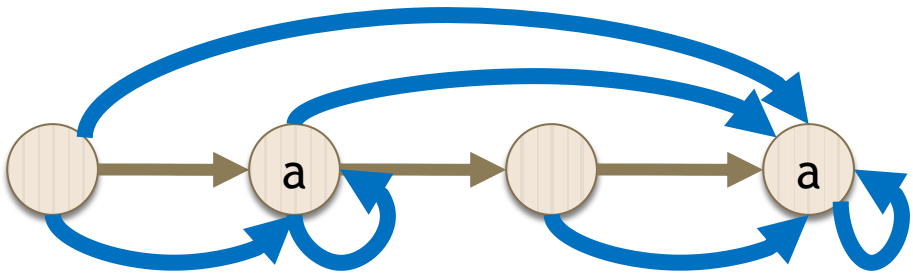
(i.e., do all existential formulas have logically-equivalent forms in this syntax?)

# More Examples

- $edge_{OUT} \equiv X$

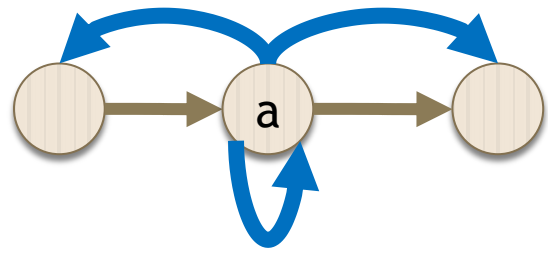


- $edge_{OUT} \equiv \mu Y. ((X \wedge a) \vee \Diamond Y)$



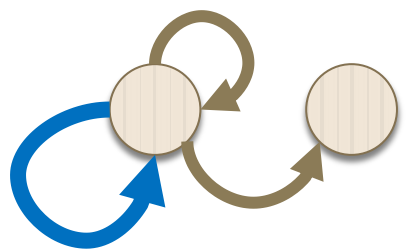
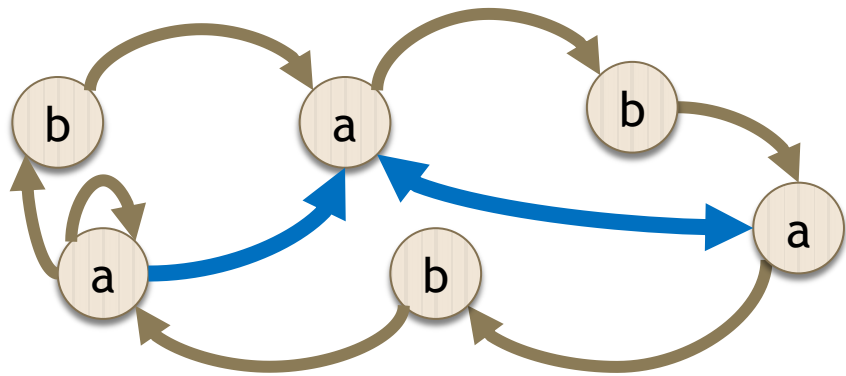
(Non-Examples)

$edge_{OUT} \equiv a$



$edge_{OUT} \equiv X \wedge \Diamond X$

- $edge_{OUT} \equiv \mu Y. ((X \wedge a \wedge \Box b) \vee (\neg a \wedge \Diamond Y))$



# Pre-Image Computation

For  $T = (\sigma_{\text{OUT}}, e_{\text{OUT}})$ , define

- $\text{inv}(\text{False}) = \text{False}$
- $\text{inv}(\neg \varphi) = \neg \text{inv}(\varphi)$
- $\text{inv}(\varphi_1 \vee \varphi_2) = \text{inv}(\varphi_1) \vee \text{inv}(\varphi_2)$
- $\text{inv}(\sigma) = \sigma_{\text{OUT}}$
- $\text{inv}(\diamond \varphi) = \text{edge}_{\text{OUT}} [X / \text{inv}(\varphi)]$
- $\text{inv}(Y) = Y$
- $\text{inv}(\mu Y. \varphi) = \mu Y. \text{inv}(\varphi)$

**Theorem:**  $f_T(G), v \models \varphi$  iff  $G, v \models \text{inv}(\varphi)$

# Proof of the Theorem

**Theorem:**  $f_T(G), v \models \varphi$  iff  $G, v \models \text{inv}(\varphi)$

- By Induction on  $\varphi$ . The essential case is:

$G, v \models \text{inv}(\diamond\varphi)$

iff  $G, v \models \text{edge}_{\text{OUT}} [X / \text{inv}(\varphi)]$  (definition of  $\text{inv}$ )

iff  $\exists w (G, v, X:\{w\} \models \text{edge}_{\text{OUT}}$  and  $G, w \models \text{inv}(\varphi))$  (ext)

iff  $\exists w ((v, w) \in E'$  and  $G, w \models \text{inv}(\varphi))$  (def of  $E'$ )

iff  $\exists w ((v, w) \in E'$  and  $f_T(G), w \models \varphi$ ) (IH)

iff  $f_T(G), v \models \diamond\varphi$  (definition of  $\diamond$ )

# n-copying

## Modal- $\mu$ Definable Transduction

A set of Modal- $\mu$  formulas  $T =$

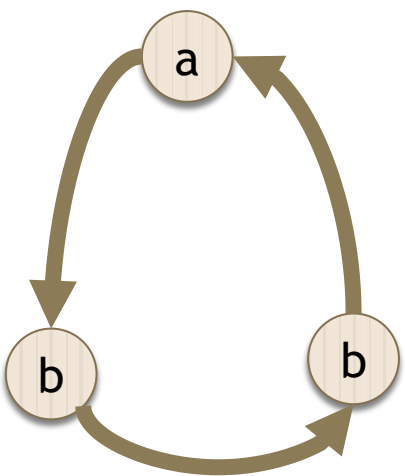
- $\sigma^k_{\text{OUT}}$  for each  $\sigma \in \Sigma$ ,  $k \in \{1 \dots n\}$
- $\text{edge}^{ik}_{\text{OUT}}$  for each  $i, k \in \{1 \dots n\}$  : *existential*

defines a transformation  $f_T$  converting

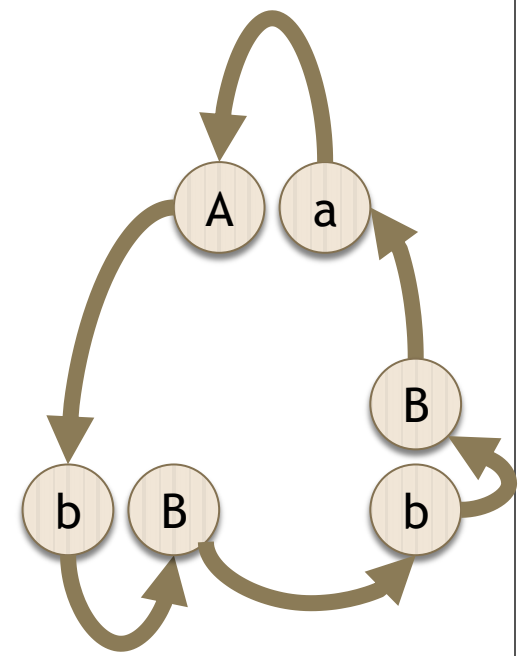
$G = (V, E, \pi)$  into  $G' = (V^*\{1..n\}, E', \pi')$  where

- $\pi'(\langle v, k \rangle) = \{ \sigma \mid G, v \models \sigma^k_{\text{OUT}} \}$
- $E' = \{ (\langle v, i \rangle, \langle w, k \rangle) \mid G, v, X:\{w\} \models \text{edge}^{ik}_{\text{OUT}} \}$

# Example ( $\Sigma = \{a, b, A, B\}$ )



$edge^{12}_{OUT} \equiv X$   
 $edge^{21}_{OUT} \equiv \diamond X$   
 $a^1_{OUT} \equiv A^2_{OUT} \equiv a$   
 $b^1_{OUT} \equiv B^2_{OUT} \equiv b$   
 otherwise  $\equiv False$



$u \vDash a \rightarrow \diamond b$

$u \vDash A^2_{OUT} \rightarrow$   
 $( edge^{21}_{OUT} [b^1_{OUT}]$   
 $\vee edge^{22}_{OUT} [b^2_{OUT}] )$

$\langle u, 2 \rangle \vDash A \rightarrow \diamond b$

# Example

- **Mutual structural recursion** (without accumulating parameters) can be dealt with.
  - For the detail of structural recursion over graphs, see [Buneman, Fernandez & Suciu 00]

- $\text{fun ev}(a \rightarrow x) = \overset{1}{A} \rightarrow \overset{2}{A} \rightarrow \text{od}(x)$
- $\text{fun ev}(b \rightarrow x) = \overset{1}{B} \rightarrow \text{od}(x)$
- $\text{fun od}(a \rightarrow x) = \overset{3}{A} \rightarrow \text{ev}(x)$
- $\text{fun od}(b \rightarrow x) = \overset{3}{B} \rightarrow \overset{4}{B} \rightarrow \text{ev}(x)$

$$\text{edge}_{\text{OUT}}^{12} \equiv a \wedge X \quad \text{edge}_{\text{OUT}}^{23} \equiv a \wedge \diamond X$$

$$\text{edge}_{\text{OUT}}^{13} \equiv a \wedge \diamond X$$

$$\text{edge}_{\text{OUT}}^{31} \equiv b \wedge \diamond X$$

$$\text{edge}_{\text{OUT}}^{34} \equiv b \wedge X \quad \text{edge}_{\text{OUT}}^{41} \equiv b \wedge \diamond X$$

# Pre-Image Computation

- $\text{inv}_k(\text{False}, \Delta) = \text{False}$
- $\text{inv}_k(\neg\varphi, \Delta) = \neg \text{inv}_k(\varphi, \Delta)$
- $\text{inv}_k(\varphi_1 \vee \varphi_2, \Delta) = \text{inv}_k(\varphi_1, \Delta) \vee \text{inv}_k(\varphi_2, \Delta)$
- $\text{inv}_k(\sigma, \Delta) = \sigma^k_{\text{OUT}}$
- $\text{inv}_k(\diamond\varphi, \Delta) = \bigvee_{j \in \{1..n\}} \text{edge}^{kj}_{\text{OUT}} [X / \text{inv}_j(\varphi, \Delta)]$
- $\text{inv}_k(Y, \Delta) = Y_k$  if  $k \in S$
- $\text{inv}_k(Y, \Delta) = \mu Y_k. \text{inv}_k(\varphi, \Delta[Y \rightarrow \langle S \cup \{k\}, \varphi \rangle])$   
where  $(S, \varphi) = \Delta(Y)$
- $\text{inv}_k(\mu Y. \varphi, \Delta) = \mu Y_k. \text{inv}_k(\varphi, \Delta[Y \rightarrow \langle \{k\}, \varphi \rangle])$

**Thm:**  $f_T(G), \langle v, k \rangle \models \varphi$  iff  $G, v \models \text{inv}_k(\varphi, \{\})$

# Example

$$\text{edge}^{11}_{\text{OUT}} \equiv \text{edge}^{12}_{\text{OUT}} \equiv \text{edge}^{21}_{\text{OUT}} \equiv \text{edge}^{22}_{\text{OUT}} \equiv \diamond X$$

$$a^1_{\text{OUT}} \equiv a^2_{\text{OUT}} \equiv a$$



OPEN QUESTION: can  $\text{inv}(\mu)$  be shorter than  $(n-1)!+1$  ?

- $f(G), \langle v, 1 \rangle \vDash \mu Y. (a \wedge \diamond Y)$
- $G, v \vDash \mu Y_1. \text{inv}_1(a \wedge \diamond Y)$
- $G, v \vDash \mu Y_1. a \wedge (\diamond \text{inv}_1(Y) \vee \diamond \text{inv}_2(Y))$
- $G, v \vDash \mu Y_1. a \wedge (\diamond Y_1 \vee \diamond \mu Y_2. \text{inv}_2(a \wedge \diamond Y))$
- $G, v \vDash \mu Y_1. a \wedge (\diamond Y_1 \vee \diamond \mu Y_2. a \wedge (\diamond \text{inv}_1(Y) \vee \diamond \text{inv}_2(Y)))$
- $G, v \vDash \mu Y_1. a \wedge (\diamond Y_1 \vee \diamond \mu Y_2. a \wedge (\diamond Y_1 \vee \diamond Y_2))$

# Some Useful Results

**Theorem:**  
**Modal- $\mu$  Definable Transduction is closed under composition.**

Construction is analogous to  $\text{inv}(\varphi)$ .

**Theorem:**  
**Modal- $\mu$  Definable Transduction**  
 **$\Leftrightarrow$  MSO Definable & Bisimulation-Invariant.**

It is known that Bisimulation-Invariant MSO transduction is equal to structural recursion [Colcombet & Löding 04].

# Conclusion

- Modal- $\mu$  Definable Transduction
  - Pre-Image of a modal- $\mu$  sentence is computable
  - Structural recursion is expressible
  - (Not in the talk)
    - Node-erasing transformations
    - Edge-labeled graphs
    - Transformations with multiple inputs/outputs
- Future Work
  - Implementation
  - Addition of **Backward Modality**
    - $(G, v \models \blacklozenge \varphi \text{ iff there's } (w, v) \in E \text{ s.t. } G, w \models \varphi)$
  - **Syntactic necessary condition for edge<sub>OUT</sub>**
  - More concise formula for **inv( $\mu Y. \varphi$ )**

# References

[Trakhtenbrot 50] **Impossibility of an Algorithm for the Decision Problem for Finite Classes**

- Satisfiability of FO on graphs is undecidable

[Meyer 74] **Weak monadic second order theory of successor is not elementary-recursive**

- Satisfiability of MSO on finite strings is Non-Elementary

[Robertson 74] **Structure of Complexity in the Weak Monadic Second-Order Theories of the Natural Numbers**

- Satisfiability of FO[<] on finite strings is Non-Elementary

[Lander 77] **The Computational Complexity of Provability in Systems of Propositional Modal Logic**

- Satisfiability of Modal Logic on graphs is PSPACE-complete

[Emerson & Jutla 88] **The Complexity of Tree Automata and Logics of Programs**

- Satisfiability of Modal- $\mu$  on graphs is EXPTIME-complete

[van Benthem 86] **Essays in Logical Semantics**

- $FO \cap \text{Bisim} = \text{Modal}$

[Janin & Walukiewicz 96] **On the Expressive Completeness of the Propositional mu-Calculus with Respect to Monadic Second Order Logic**

- $MSO \cap \text{Bisim} = \text{Modal-}\mu$

[Colcombet & Löding 04] **On the Expressiveness of Deterministic Transducers over Infinite Trees**

- $MSO\text{-Definable Graph Transduction} \cap \text{Bisim} = \text{Structural Recursion}$

[Courcelle 94] **Monadic Second-Order Definable Graph Transductions: A Survey**

- On MSO-Definable Transduction