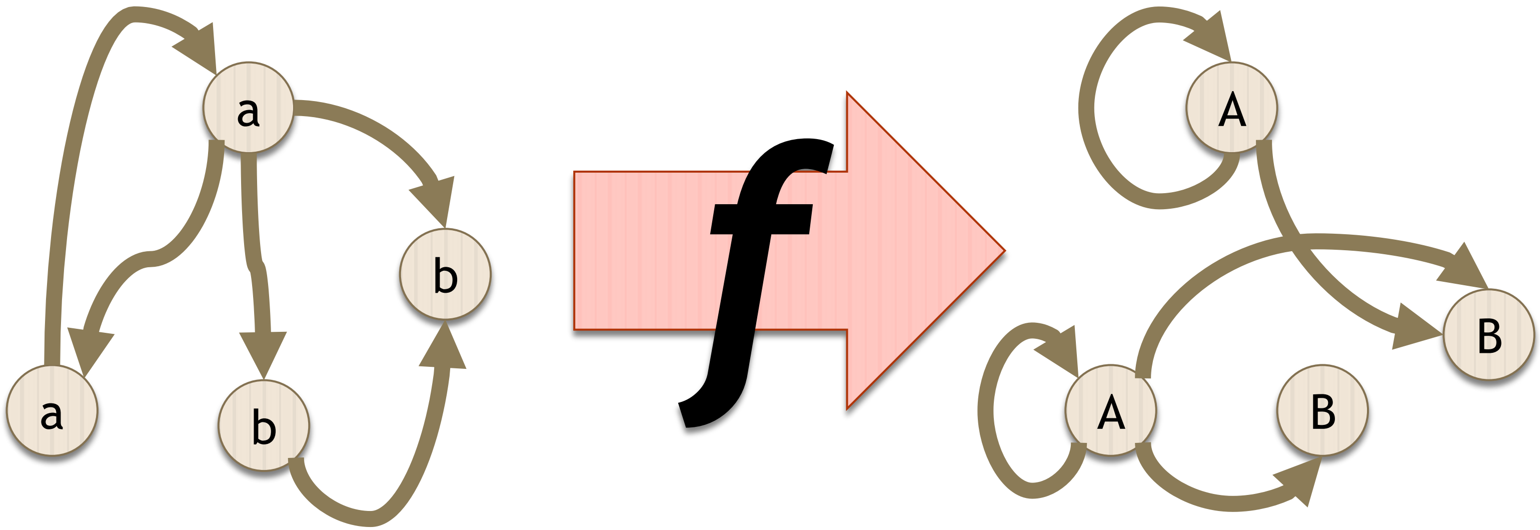
Modal-μ Definable Graph Transduction

Kazuhiro Inaba
National Institute of Informatics, Japan

4th DIKU-IST Workshop, 2011

What We Want to Do

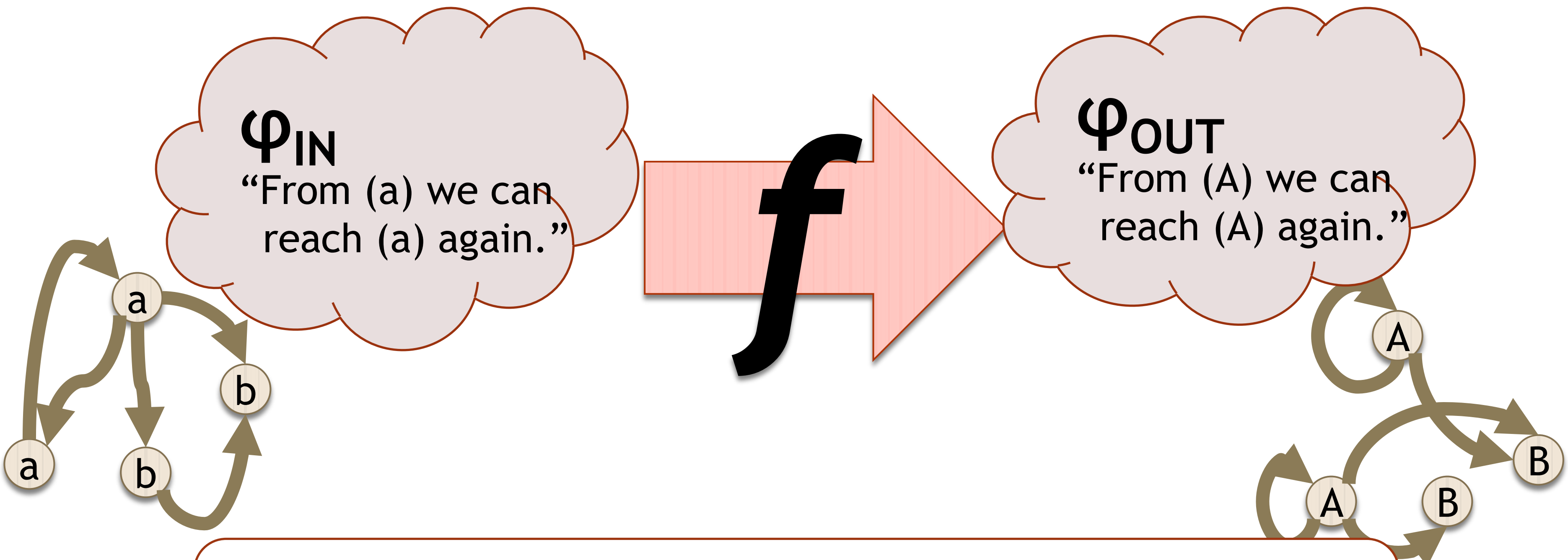
- Verification of **Graph-to-Graph Transformations**



- 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**



verify whether or not:

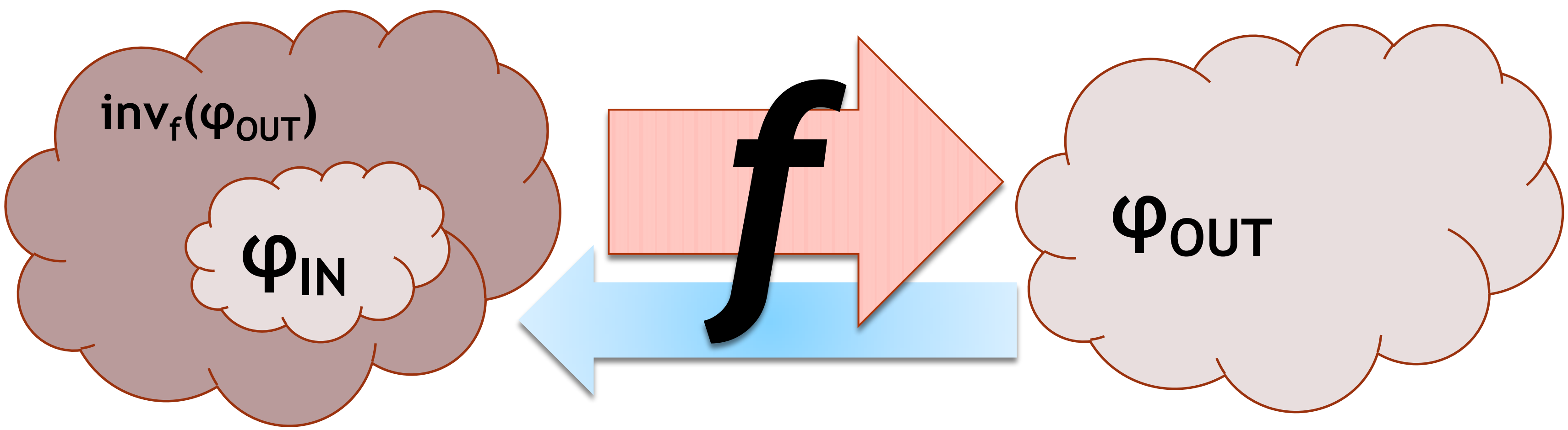
for any graph G, $G \models \varphi_{IN} \Rightarrow f(G) \models \varphi_{OUT}$

Verification by Pre-Image

(a.k.a. "weakest precondition" or "inverse type inference")

Given f and φ_{OUT} , compute $inv_f(\varphi_{OUT})$ such that:

for any graph G, $f(G) \vDash \varphi_{OUT}$ iff $G \vDash inv_f(\varphi_{OUT})$



Then **"for any graph G, $G \vDash \varphi_{IN} \Rightarrow f(G) \vDash \varphi_{OUT}$"**

iff **"for any graph G, $G \vDash (\varphi_{IN} \rightarrow inv_f(\varphi_{OUT}))$"**

i.e., $\varphi_{IN} \rightarrow inv_f(\varphi_{OUT})$ is *valid*

To Be More Concrete…

- Which logic can we use for specifying $\varphi_{IN/OUT}$?
  - 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-μ Calculus** as the specification logic
  - (At least for trees) capture all existing XML-Schemas
- Define a new model of graph transformation called ***Modal-μ Definable Transduction***
  - Pre-image of modal-μ sentence can be fully automatically computed
  - Expressive enough to capture (unnested) structural recursion on graphs

Related Work

- MSO (Monadic 2nd-Order Logic) Definable Transduction
  - Overall structure is more or less the same.
  - Ours is a proposal to use **Modal-μ 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)
{ φIN }   f   { φOUT }
```

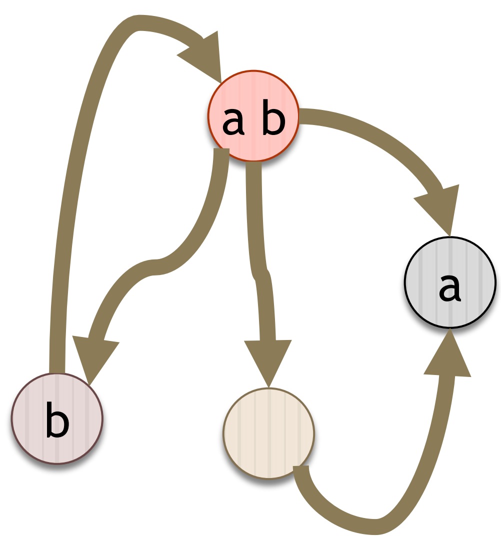
```
{ φIN }
  p := root
  while  p != null do
    q := p.next
    p.next := p.next.next
    p := q
  end
{ φOUT }
```

Outline

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

Graphs (in Today's Talk)

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



=

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

$V = \{$ (a b), (a), (b), ( ) $\}$

$\pi =$

a b $\longrightarrow \{a,b\}$

a $\longrightarrow \{a\}$

b $\longrightarrow \{b\}$

$\longrightarrow \{\}$

Predicate Logics on Graphs

FO

φ ::=

| False | $\neg\, \varphi$ | $\varphi \lor \varphi$

| $\sigma(x)$ (for $\sigma \in \Sigma$) "node x is labeled σ"

| edge(x, y) "an edge connects x to y"

| $\exists x.\ \varphi$ "there's x that makes ψ hold"

| $\exists S.\ \varphi$ "there's a set S that makes ψ hold"

| $x \in S$ "x is in S"

MSO

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

Semantics

- For a graph $G=(V,E,\pi)$ and an environment $\Gamma : \mathrm{Var} \to V$

  - **$G, \Gamma \models \sigma(x)$** iff $\sigma \in \pi(\Gamma(x))$
    "node x is labeled σ"
  - **$G, \Gamma \models \mathrm{edge}(x, y)$** 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

ψ ::=

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

| σ (for $\sigma \in \Sigma$) "current node is labeled σ"

| $\Diamond\varphi$ "current node has an outgoing edge whose destination satisfies φ"

M

| X

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

Mμ

We Can Define: $\Box\varphi$ (dual of $\Diamond$) and $\nu X.\varphi$ (GreatestFixPt)

Semantics

- For a graph $G=(V,E,\pi)$, an environment $\Gamma : Var \rightarrow 2^V$, and the current node $v \in V$

- $\mathbf{G, v, \Gamma \vDash \sigma}$ iff $\sigma \in \pi(v)$

  "current node is labeled σ"

- $\mathbf{G, v, \Gamma \vDash \Diamond\varphi}$ iff there's w $(v,w)\in E$ & $G,w,\Gamma \vDash \varphi$

  "current node has an outgoing edge whose destination satisfies φ"

- $\mathbf{G, v, \Gamma \vDash \mu Y.\ \varphi}$ iff $v \in LFP(F)$

  where $F(A) = \{w\in V \mid G, w, \Gamma[Y:A] \vDash \varphi\}$

…

Examples

- "From the node x, we can reach a σ-node"

$$\forall S.\ (\ (x \in S \land \forall y. \forall z. (y \in S \land (edge(y,z) \rightarrow z \in S))) \rightarrow \exists y.\ (y \in S \land \sigma(y)))$$

- "Confluence"

$$\forall y.\ \forall z.\ (\ edge(x,y) \land edge(x,z) \rightarrow \exists w.\ (edge(y,w) \land edge(z,w))\ )$$

- "From the current node, we can reach a σ-node"

$$\mu Y.\ (\sigma \lor \Diamond Y)$$

- "Confluence"

(No way to express it in Modal-μ)

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

A set of MSO formulas **T** =

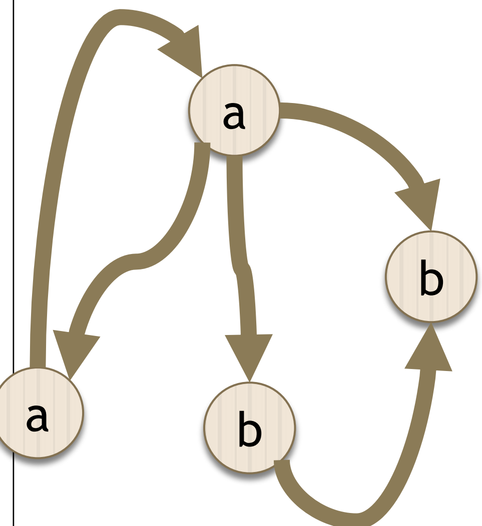
- $\sigma_{OUT}(x)$ for each $\sigma \in \Sigma$
- $\mathbf{edge}_{OUT}(x,y)$

defines a transformation f_T converting

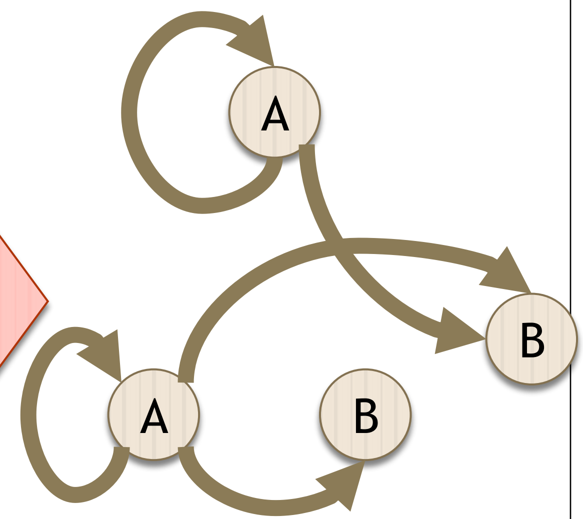
G = (V, E, π) into **G' = (V, E', π')** where

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

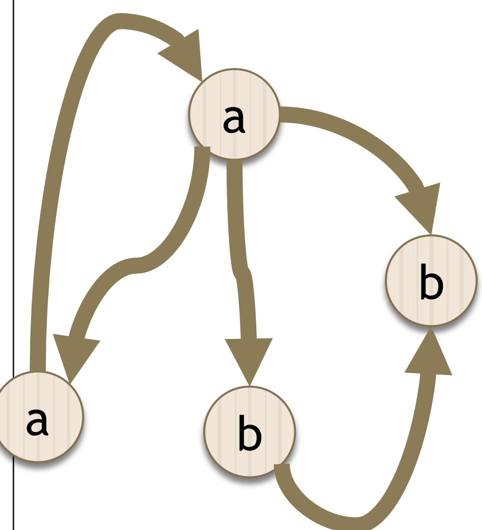
Example ($\Sigma = \{a, b, A, B\}$)



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



Pre-Image is Easily Obtained



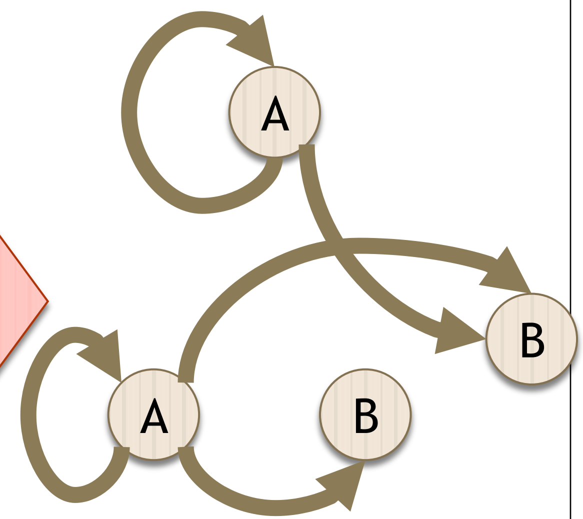
$\mathrm{edge}_{OUT}(x, y) \equiv$

$\exists z.(\mathrm{edge}(x,z) \wedge \mathrm{edge}(z,y))$

$a_{OUT}(x) \equiv b_{OUT}(x) \equiv \mathrm{False}$

$A_{OUT}(x) \equiv a(x)$

$B_{OUT}(x) \equiv b(x)$

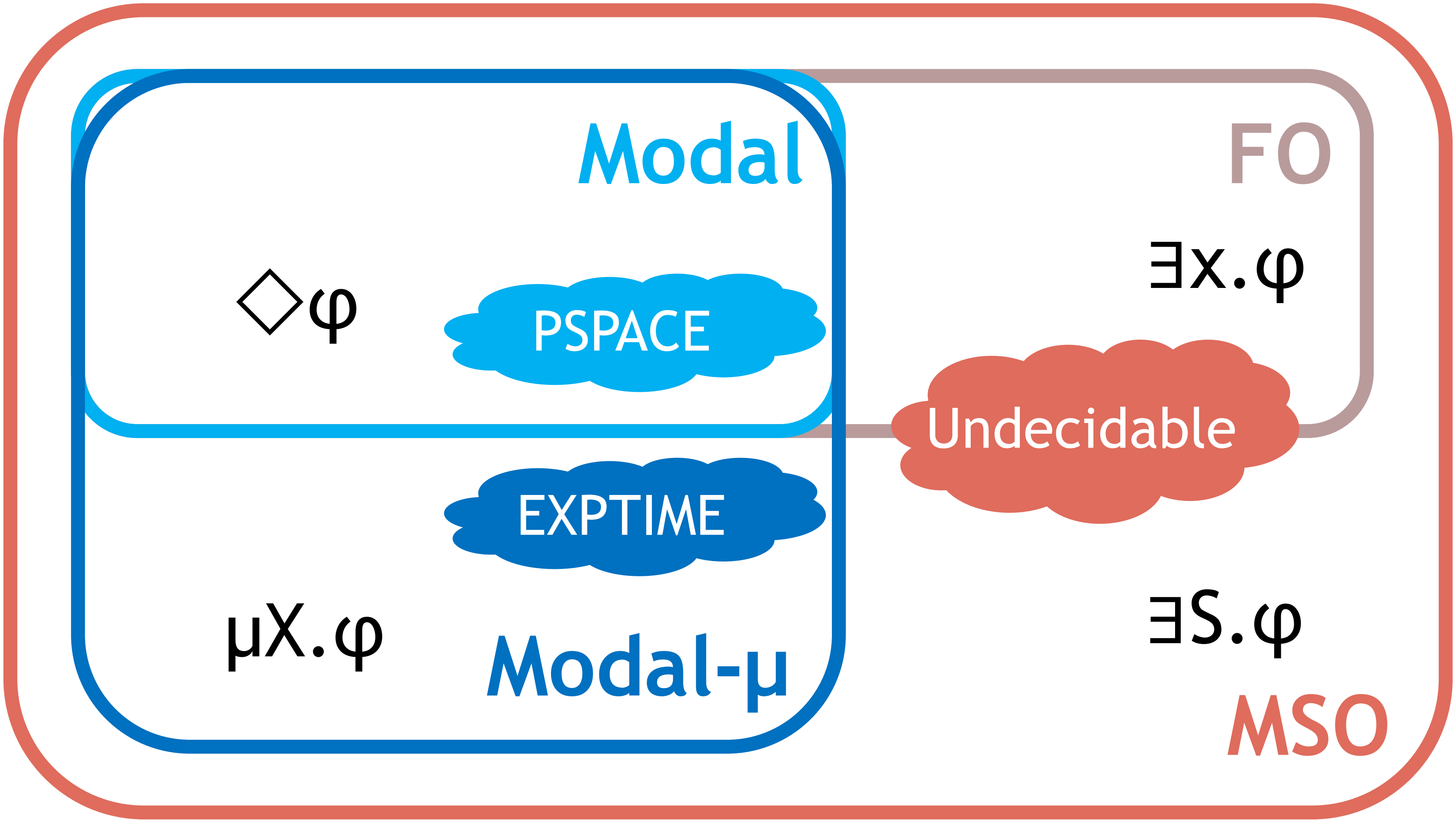


$\forall x.\ a(x) \rightarrow \exists y.$
$\exists z.(\mathrm{edge}(x,z) \wedge \mathrm{edge}(z,y))$
$\wedge\ a(y)$

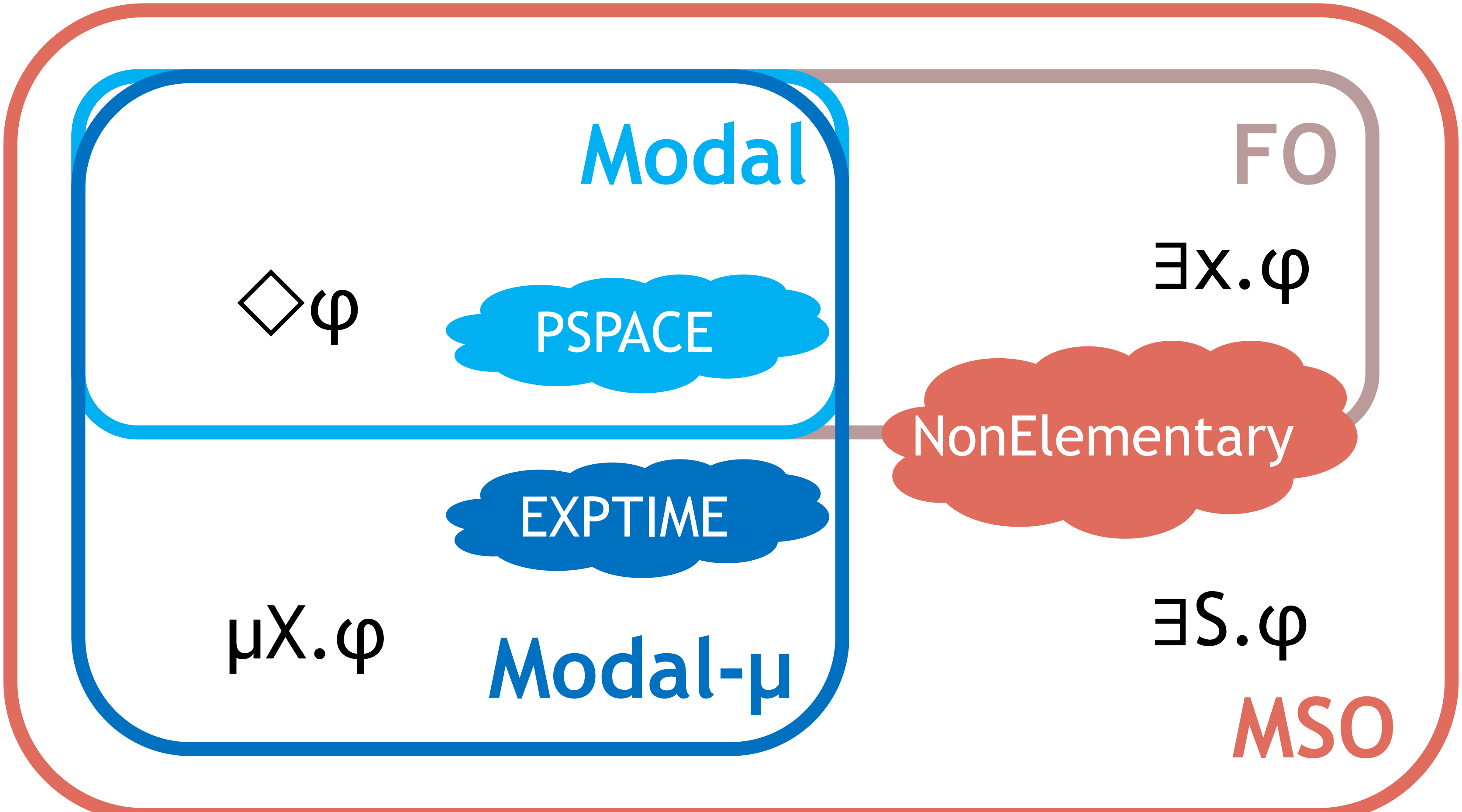
Inline Expansion

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

Expressiveness & Complexity



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



Modal-μ and MSO

- Complexity of Validity Checking
  - **Modal-μ : EXPTIME-complete**
  - MSO : Undecidable (Even in Trees, HyperEXP)

- Expressive Power
  - Modal-μ = ***Bisimulation-Invariant*** **Subset of MSO** [Janin & Walukiewicz 96]
  - "Bisimulation-Invariant" ≃ "Physical equality of pointers cannot be checked"
  - Not a severe restriction for purely functional programs!

Modal-μ Definable (1-copying) Transduction

A set of Modal-μ formulas T =

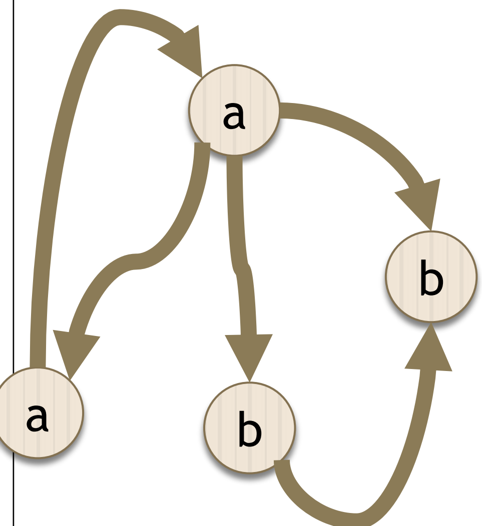
- σ_{OUT} for each $\sigma \in \Sigma$
- $edge_{OUT}$ an ***existential*** formula Fv={X}

defines a transformation $\mathbf{f_T}$ converting

G = (V, E, π) into **G' = (V, E', π')** where

- $\pi'(\ v\ ) = \{\ \sigma \mid G, v \vDash \sigma_{OUT}\ \}$
- $E' = \{\ (v, w) \mid G, v, X{:}\{w\} \vDash edge_{OUT}\ \}$

Example (Σ = {a, b, A, B})

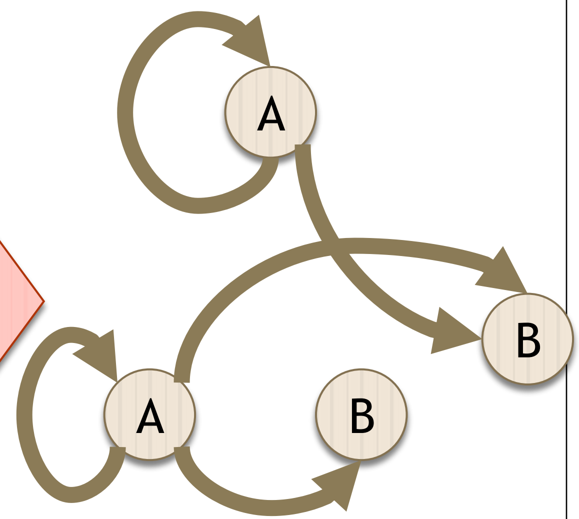


$\mathbf{edge}_{OUT} \equiv \Diamond\Diamond X$

$a_{OUT} \equiv b_{OUT} \equiv \mathrm{False}$

$A_{OUT} \equiv a$

$B_{OUT} \equiv b$



$a \rightarrow \Diamond\Diamond a$

$A \rightarrow \Diamond A$

Existential Formula

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

for all G=(V,E,π), v∈V, P⊆V
G, v, X:P ⊨ e iff ∃w∈P. G, v, X:{w} ⊨ e

- Examples:
  - "X ∨ True" is not existential (Consider P={}).
  - "◇X" is existential.
  - "□X" is not (when v is a leaf node ...).
  - "σ" is not, but "X ∧ σ" 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

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

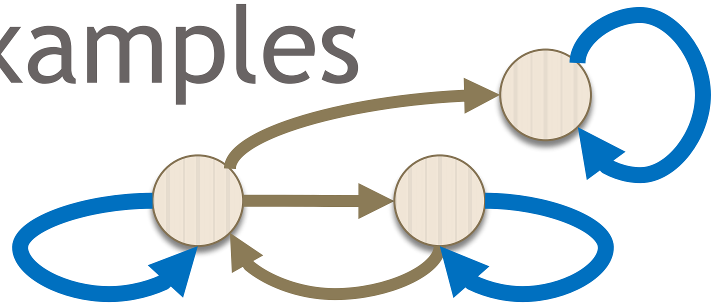
(True, $\neg$, σ, $\Box$, and GFP must be "guarded" by _ $\wedge$ _)

OPEN QUESTION: is this a necessary condition ?

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

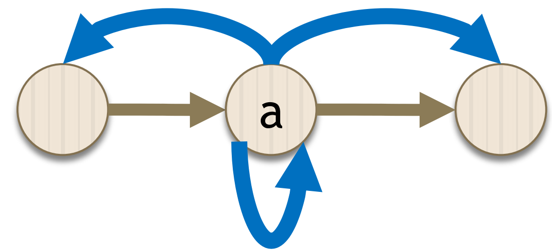
More Examples

- $\text{edge}_{\text{OUT}} \equiv X$

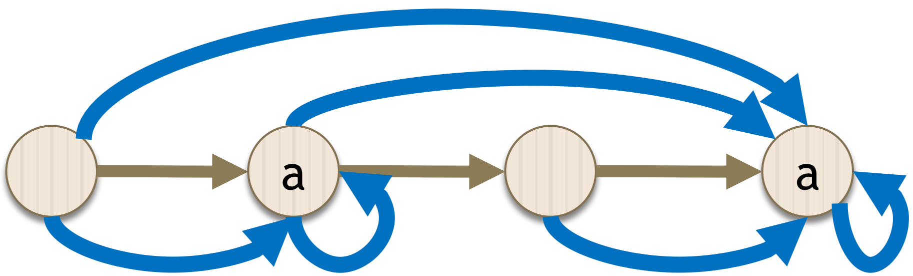


(Non-Examples)

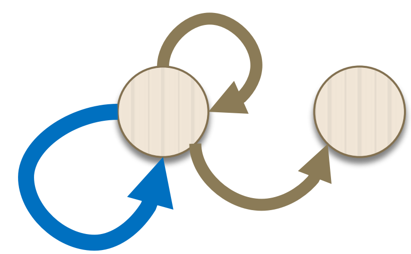
$\text{edge}_{\text{OUT}} \equiv a$



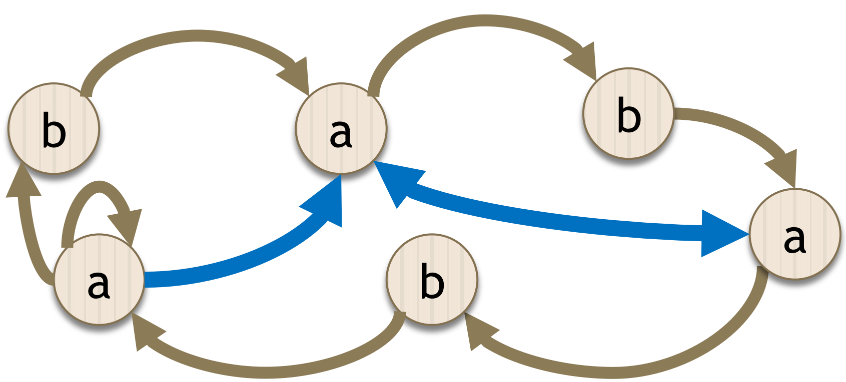
- $\text{edge}_{\text{OUT}} \equiv \mu Y.\ ((X \wedge a) \vee \Diamond Y)$



$\text{edge}_{\text{OUT}} \equiv X \wedge \Diamond X$



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



Pre-Image Computation

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

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

Theorem: $f_T(G), v \models \varphi$ iff $G, v \models inv(\varphi)$

Proof of the Theorem

Theorem: $f_T(G), v \models \varphi$ iff $G, v \models inv(\varphi)$

- By Induction on φ. The essential case is:

$G, v \models inv(\Diamond\varphi)$

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

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

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

n-copying Modal-μ Definable Transduction

A set of Modal-μ formulas T =

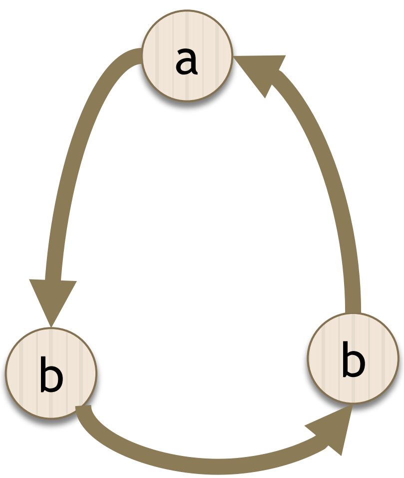
- σ^k_{OUT} for each $\sigma \in \Sigma$, $k \in \{1 .. n\}$
- $edge^{ik}_{OUT}$ for each i, $k \in \{1 .. n\}$: ***existential***

defines a transformation f_T converting

G = (V, E, π) into **G' = (V*{1..n}, E', π')** where

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

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



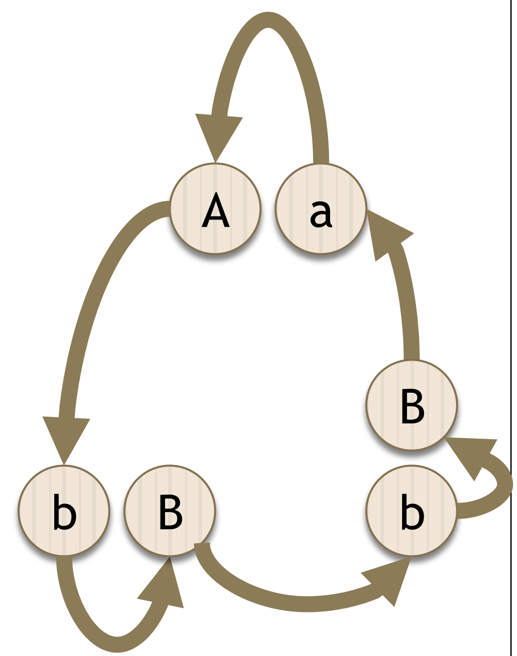
$\mathbf{edge^{12}_{OUT} \equiv X}$

$\mathbf{edge^{21}_{OUT} \equiv \Diamond X}$

$\mathbf{a^1_{OUT} \equiv A^2_{OUT} \equiv a}$

$\mathbf{b^1_{OUT} \equiv B^2_{OUT} \equiv b}$

otherwise ≡ False



$u \models a \rightarrow \Diamond b$

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

$\mathbf{\langle u,2 \rangle \models A \rightarrow \Diamond b}$

Example

- **Mutual structural recursion** (without accumulating parameters) can be dealt with.
  - For the detail of structural recursion over graphs, see [Buneman, Fernandez & Suciu 00]
- fun ev($a \to$ x) = $^{1}A \to {}^{2}A \to$ od(x)
- fun ev($b \to$ x) = $^{1}B \to$ od(x)
- fun od($a \to$ x) = $^{3}A \to$ ev(x)
- fun od($b \to$ x) = $^{3}B \to {}^{4}B \to$ ev(x)

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

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

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

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

Pre-Image Computation

- inv_k (False, Δ) = False
- $inv_k (\neg\varphi , \Delta) = \neg\, inv_k (\varphi, \Delta)$
- $inv_k (\varphi_1 \vee \varphi_2, \Delta) = inv_k (\varphi_1, \Delta) \vee inv_k (\varphi_2, \Delta)$
- $inv_k (\sigma, \Delta) = \sigma^k{}_{OUT}$
- $inv_k (\Diamond\varphi, \Delta) = \bigvee_{j\in\{1..n\}} edge^{kj}{}_{OUT} [X / inv_j(\varphi, \Delta)]$
- $inv_k (Y, \Delta) = Y_k$ if $k \in S$
- $inv_k (Y, \Delta) = \mu Y_k.\ inv_k(\varphi, \Delta[Y \rightarrow <S \cup \{k\},\varphi>])$ where $(S,\varphi) = \Delta(Y)$
- $inv_k (\mu Y.\varphi , \Delta) = \mu Y_k.\ inv_k(\varphi, \Delta[Y \rightarrow <\{k\},\varphi>])$

Thm: $f_T(G), <v,k> \models \varphi$ iff $G,v \models inv_k(\varphi, \{\})$

Example

$\mathrm{edge}^{11}_{OUT} \equiv \mathrm{edge}^{12}_{OUT} \equiv \mathrm{edge}^{21}_{OUT} \equiv \mathrm{edge}^{22}_{OUT} \equiv \Diamond X$

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

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

- $f(G), \langle v,1 \rangle \models \mu Y.\ (a \wedge \Diamond Y)$
- $G, v \models \mu Y_1.\ inv_1(\ a \wedge \Diamond Y\ )$
- $G, v \models \mu Y_1.\ \ a \wedge (\Diamond inv_1(Y) \vee \Diamond inv_2(Y))$
- $G, v \models \mu Y_1.\ \ a \wedge (\Diamond Y_1 \vee \Diamond \mu Y_2. inv_2(a \wedge \Diamond Y))$
- $G, v \models \mu Y_1.\ a \wedge (\Diamond Y_1 \vee \Diamond \mu Y_2.\ a \wedge (\Diamond inv_1(Y) \vee \Diamond inv_2(Y))$
- $G, v \models \mu Y_1.\ a \wedge (\Diamond Y_1 \vee \Diamond \mu Y_2.\ a \wedge (\Diamond Y_1 \vee \Diamond Y_2))$

Some Useful Results

Theorem:
Modal-μ Definable Transduction is closed under composition.

Construction is analogous to inv(φ).

Theorem:
Modal-μ Definable Transduction
⇔ MSO Definable & Bisimulation-Invariant.

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

Conclusion

- Modal-μ Definable Transduction
  - Pre-Image of a modal-μ 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**
    - ($G,v \models \blacklozenge\varphi$ iff there's $(w,v)\in E$ s.t. $G,w \models \varphi$)
  - **Syntactic necessary condition for edge_{OUT}**
  - 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

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

- Satisiability of Modal-μ on graphs is EXPTIME-complete

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

- FO ∩ Bisim = Modal

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

- MSO ∩ Bisim = Modal-μ

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

- MSO-Definable Graph Transduction ∩ Bisim = Structural Recursion

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

- On MSO-Definable Transduction