Modal-\(\mu\) 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

\[ \varphi_{IN} \quad \text{"From (a) we can reach (a) again."} \]

\[ \varphi_{OUT} \quad \text{"From (A) we can reach (A) again."} \]

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 \( \varphi_{\text{OUT}} \), 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 $\varphi_{IN/OUT}$?
  - 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 2\textsuperscript{nd}-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

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

```plaintext
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

\[
\Sigma = \{a, b\} \\
V = \{a, b\} \\
\pi = \\
\begin{align*}
\rightarrow & \{a, b\} \\
\rightarrow & \{a\} \\
\rightarrow & \{b\} \\
\rightarrow & \{\}\n\end{align*}
\]
Predicate Logics on Graphs

\[ \varphi ::= \]

- \( \neg \varphi \)
- \( \varphi \lor \varphi \)
- \( \sigma(x) \) (for \( \sigma \in \Sigma \)) “node \( x \) is labeled \( \sigma \)”
- \( \text{edge}(x, y) \) “an edge connects \( x \) to \( y \)”
- \( \exists x. \varphi \) “there’s \( x \) that makes \( \psi \) hold”
- \( \exists S. \varphi \) “there’s a set \( S \) that makes \( \psi \) hold”
- \( x \in S \) “\( x \) is in \( S \)”

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 \( \Gamma : \text{Var} \rightarrow V \)

  - \( G, \Gamma \models \sigma(x) \) 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
- \( \neg \varphi \)
- \( \varphi \lor \varphi \)
- \( \sigma \) (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 \( \neg \))

We Can Define: \( \Box \varphi \) (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 $v \in V$

- $G, v, \Gamma \vdash \sigma$ iff $\sigma \in \pi(v)$
  “current node is labeled $\sigma$”

- $G, v, \Gamma \vdash \Diamond \varphi$ iff there’s $w$ $(v, w) \in E$ & $G, w, \Gamma \vdash \varphi$
  “current node has an outgoing edge whose destination satisfies $\varphi$”

- $G, v, \Gamma \vdash \mu Y. \varphi$ iff $v \in \text{LFP}(F)$
  where $F(A) = \{w \in V \mid G, w, \Gamma[Y:A] \vdash \varphi\}$

...
Examples

• “From the node \( x \), we can reach a \( \sigma \)-node”
  \[ \forall S. \ ( (x \in S \land \forall y. \forall z. (y \in S \land \\
  \quad (\text{edge}(y,z) \to z \in S))) \to \exists y. \ (y \in S \land \sigma(y))) \]

• “Confluence”
  \[ \forall y. \forall z. \ (\text{edge}(x,y) \land \text{edge}(x,z) \to \exists w. \ (\text{edge}(y,w) \land \text{edge}(z,w)) ) \]

• “From the current node, we can reach a \( \sigma \)-node”
  \[ \mu Y. \ (\sigma \land \Diamond Y) \]

• “Confluence”
  \[ (\text{No way to express it in Modal-\( \mu \))} \]
MSO Definable (1-copying) Transduction  
[Courcelle 94]

A set of MSO formulas $T =$
- $\sigma_{\text{OUT}}(x)$ for each $\sigma \in \Sigma$
- $\text{edge}_{\text{OUT}}(x, y)$

defines a transformation $f_T$ converting
$G = (V, E, \pi)$ into $G' = (V, E', \pi')$ where

- $\pi'(v) = \{ \sigma \mid G, x:v \models \sigma_{\text{OUT}}(x) \}$
- $E' = \{ (v, w) \mid G, x:v, y:w \models \text{edge}_{\text{OUT}}(x, y) \}$
Example ($\Sigma = \{a, b, A, B\}$)

\[
\text{edge}_{\text{OUT}}(x, y) \equiv \\
\exists z. (\text{edge}(x, z) \land \text{edge}(z, y))
\]

\[
a_{\text{OUT}}(x) \equiv b_{\text{OUT}}(x) \equiv \text{False}
\]

\[
A_{\text{OUT}}(x) \equiv a(x)
\]

\[
B_{\text{OUT}}(x) \equiv b(x)
\]
Pre-Image is Easily Obtained

\[ \forall x. \ a(x) \rightarrow \exists y. \ \exists z. (\text{edge}(x,z) \land \text{edge}(z,y)) \land a(y) \]

\[ \forall x. \ \text{A}(x) \rightarrow \exists y. \ \text{edge}(x,y) \land \text{A}(y) \]

\[ \text{edge}_{\text{OUT}}(x, y) \equiv \exists z. (\text{edge}(x,z) \land \text{edge}(z,y)) \]

\[ \text{a}_{\text{OUT}}(x) \equiv \text{b}_{\text{OUT}}(x) \equiv \text{False} \]

\[ \text{A}_{\text{OUT}}(x) \equiv a(x) \]

\[ \text{B}_{\text{OUT}}(x) \equiv b(x) \]
Expressiveness & Complexity

Modal

\( \diamond \phi \)

PSPACE

\( \mu X. \phi \)

Modal-\( \mu \)

Modal

EXPTIME

Undecidable

FO

\( \exists x. \phi \)

MSO

\( \exists S. \phi \)
Expressiveness & Complexity (on “tree-like” graphs)

FO

∃x.φ

PSPACE

∃x.φ

EXPTIME

NonElementary

∃S.φ

Modal

Modal

Modal-μ

μX.φ

◇φ

\[293x326\]

\[253x305\]

\[144x263\]

\[176x263\]

\[140x263\]

\[140x122\]

\[155x128\]

\[556x281\]

\[557x281\]

\[538x281\]

\[572x281\]

\[573x79\]

\[293x326\]

\[253x105\]

\[144x263\]

\[176x263\]

\[140x263\]

\[140x122\]

\[155x128\]

\[556x281\]

\[557x281\]

\[538x281\]

\[572x281\]

\[573x79\]
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” \(\simeq\)
    - “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 \textit{existential} formula $F_{v} = \{X\}$ 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 ($\Sigma = \{a, b, A, B\}$)

\[
\begin{align*}
\text{edge}_{\text{OUT}} & \equiv \text{◇◇X} \\
\text{a}_{\text{OUT}} & \equiv \text{b}_{\text{OUT}} \equiv \text{False} \\
\text{A}_{\text{OUT}} & \equiv \text{a} \\
\text{B}_{\text{OUT}} & \equiv \text{b}
\end{align*}
\]
Existential Formula

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

  \[
  \text{for all } G=(V,E,\pi), \ v \in V, \ P \subseteq V \quad G, \ v, \ X:P \vDash e \iff \exists w \in P. \ G, \ v, \ X:\{w\} \vDash e
  \]

- **Examples:**
  - “$X \lor \text{True}$” is not existential (Consider $P=\{\}$).
  - “$\lozenge X$” is existential.
  - “$\square X$” is not (when $v$ is a leaf node ...).
  - “$\sigma$” is not, but “$X \land \sigma$” is.
Syntactic Condition

for all $G=(V,E,\pi)$, $v \in V$, $P \subseteq V$

$G, v, X:P \vDash e \iff \exists w \in P. G, v, X:\{w\} \vDash e$

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

\[
e ::= \text{False} \mid X \mid Y \mid e \lor e \mid \Diamond e \mid \mu Y. e \\
\mid e \land \varphi \quad \text{where } \varphi \text{ is any formula without free variables}
\]

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

**Open Question:** is this a necessary condition?
(i.e., do all existential formulas have logically-equivalent forms in this syntax?)
More Examples

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

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

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

(Non-Examples)

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

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

- $\text{edge}_{\text{OUT}} \equiv X \land \Diamond X$
Pre-Image Computation

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

- $\text{inv}(\text{False}) = \text{False}$
- $\text{inv}(\neg \phi) = \neg \text{inv}(\phi)$
- $\text{inv}(\phi_1 \lor \phi_2) = \text{inv}(\phi_1) \lor \text{inv}(\phi_2)$
- $\text{inv}(\sigma) = \sigma_{OUT}$
- $\text{inv}(\Diamond \phi) = \text{edge}_{OUT}[X / \text{inv}(\phi)]$ 
- $\text{inv}(Y) = Y$
- $\text{inv}(\mu Y. \phi) = \mu Y. \text{inv}(\phi)$

Theorem: $f_T(G), v \vDash \phi$ iff $G, v \vDash \text{inv}(\phi)$
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:

  \[ G, v \not\models \text{inv}(\varphi) \]

  iff \( G, v \not\models \text{edge}_{\text{OUT}} \ [X / \text{inv}(\varphi)] \) (definition of inv)

  iff \( \exists w (G, v, X : \{w\} \not\models \text{edge}_{\text{OUT}} \text{ and } G, w \not\models \text{inv}(\varphi)) \) (ext)

  iff \( \exists w ((v, w) \in E' \text{ and } G, w \not\models \text{inv}(\varphi)) \) (def of \( E' \))

  iff \( \exists w ((v, w) \in E' \text{ and } f_T(G), w \not\models \varphi) \) (IH)

  iff \( f_T(G), v \not\models \Diamond \varphi \) (definition of \( \Diamond \))
n-copying
Modal-μ Definable Transduction

A set of Modal-μ formulas $T =$

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

defines a transformation $f_T$ converting

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

- $\pi'( <v,k> ) = \{ \sigma \mid G, v \models \sigma^k_{\text{OUT}} \}$
- $E' = \{ ( <v,i>, <w,k> ) \mid G, v, X:\{w\} \models \text{edge}^{ik}_{\text{OUT}} \}$
Example ($\Sigma = \{a, b, A, B\}$)

\[
\begin{align*}
\text{edge}^{12}_{\text{OUT}} & \equiv X \\
\text{edge}^{21}_{\text{OUT}} & \equiv \Diamond X \\
a^1_{\text{OUT}} & \equiv A^2_{\text{OUT}} \equiv a \\
b^1_{\text{OUT}} & \equiv B^2_{\text{OUT}} \equiv b \\
\text{otherwise} & \equiv \text{False}
\end{align*}
\]
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 → x) = od(x)`
- `fun ev(b → x) = od(x)`
- `fun od(a → x) = ev(x)`
- `fun od(b → x) = ev(x)`

\[
\begin{align*}
\text{edge}_{12}^{\text{OUT}} & \equiv a \land X \\
\text{edge}_{23}^{\text{OUT}} & \equiv a \land \Diamond X \\
\text{edge}_{13}^{\text{OUT}} & \equiv a \land \Diamond X \\
\text{edge}_{31}^{\text{OUT}} & \equiv b \land \Diamond X \\
\text{edge}_{34}^{\text{OUT}} & \equiv b \land X \\
\text{edge}_{41}^{\text{OUT}} & \equiv b \land \Diamond X
\end{align*}
\]
Pre-Image Computation

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

Thm: \( f_T(G), <v,k> \models \varphi \) iff \( G,v \models \text{inv}_k(\varphi, \{\} ) \)
Example

\[ \text{edge}^{11}_\text{OUT} \equiv \text{edge}^{12}_\text{OUT} \equiv \text{edge}^{21}_\text{OUT} \equiv \text{edge}^{22}_\text{OUT} \equiv \Box X \]
\[ a^1_\text{OUT} \equiv a^2_\text{OUT} \equiv a \]

- \( f(G), <v,1> \models \mu Y. (a \land \Box Y) \)
- \( G, v \models \mu Y_1. \; \text{inv}_1(a \land \Box Y) \)
- \( G, v \models \mu Y_1. \; a \land (\Box \text{inv}_1(Y) \lor \Box \text{inv}_2(Y)) \)
- \( G, v \models \mu Y_1. \; a \land (\Box Y_1 \lor \mu Y_2. \text{inv}_2(a \land \Box Y)) \)
- \( G, v \models \mu Y_1. \; a \land (\Box Y_1 \lor \mu Y_2. \; a \land (\Box \text{inv}_1(Y) \lor \Box \text{inv}_2(Y))) \)
- \( G, v \models \mu Y_1. \; a \land (\Box Y_1 \lor \mu Y_2. \; a \land (\Box Y_1 \lor \Box Y_2)) \)

**Open Question:** can \( \text{inv}(\mu) \) be shorter than \((n-1)!+1\)?
Some Useful Results

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

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

Theorem:
**Modal-μ 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
    - \((G,v \vdash \Box \phi \iff \text{there's } (w,v) \in E \text{ s.t. } G,w \not\vdash \phi)\)
  - Syntactic necessary condition for \(\text{edge}_{\text{OUT}}\)
  - More concise formula for \(\text{inv}(\mu Y.\phi)\)
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

- Satisfiability of Modal Logic on graphs is PSPACE-complete

[Emereson & 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 $\cap$ Bisim = Modal

[Janin & Walukiewicz 96] On the Expressive Completeness of the Propositional mu-Calculus with Respect to Monadic Second Order Logic
- MSO $\cap$ Bisim = Modal-$\mu$

[Colcombet & Löding 04] On the Expressiveness of Deterministic Transducers over Infinite Trees
- MSO-Definable Graph Transduction $\cap$ Bisim = Structural Recursion

[Courcelle 94] Monadic Second-Order Definable Graph Transductions: A Survey
- On MSO-Definable Transduction