A decision procedure for alternation-free modal μ-calculi

Yoshinori Tanabe[1] Koichi Takahashi[2] Masami Hagiya[1]

1: University of Tokyo

2: National Institute of Advanced Industrial Science and Technology

Advanced in Modal Logic 2008

Motivation

Analysis of variants of graph rewriting systems
using decision procedures for satisfiability of temporal/modal logics

Previous work:

- Decision Procedure for two-way CTL [Hagiya et al 04]
  - Applied to analysis of one-dimensional cellular automata.
- Decision Procedure for alternation-free two-way μ-calculus [Tanabe et al 05]
  - Applied to analysis of XML transducers.

This research:

- Decision Procedure: (sublogics of) alternation-free μ + nominals + backward modalities + functional modalities.
  (variants of enriched μ-calculus)
- Target: analysis of programs that manipulates pointers.

Motivation

Analysis of variants of graph rewriting systems
using decision procedures for satisfiability of temporal/modal logics
Previous work:

- Decision Procedure for two-way CTL [Hagiya et al 04]
  - Applied to analysis of one-dimensional cellular automata.
- Decision Procedure for alternation-free two-way μ-calculus [Tanabe et al 05]
  - Applied to analysis of XML transducers.

This research:

- Decision Procedure: (sublogics of) alternation-free μ + nominals + backward modalities + functional modalities.
  (variants of enriched μ-calculus)
- Target: analysis of programs that manipulates pointers.

Motivation

Analysis of variants of graph rewriting systems
using decision procedures for satisfiability of temporal/modal logics
Previous work:

- Decision Procedure for two-way CTL [Hagiya et al 04]
  - Applied to analysis of one-dimensional cellular automata.
- Decision Procedure for alternation-free two-way μ-calculus [Tanabe et al 05]
  - Applied to analysis of XML transducers.

This research:

- Decision Procedure: (sublogics of) alternation-free μ + nominals + backward modalities + functional modalities.
  (variants of enriched μ-calculus)
- Target: analysis of programs that manipulates pointers.

Motivation

Analysis of variants of graph rewriting systems
using decision procedures for satisfiability of temporal/modal logics
Previous work:

- Decision Procedure for two-way CTL [Hagiya et al 04]
  - Applied to analysis of one-dimensional cellular automata.
- Decision Procedure for alternation-free two-way μ-calculus [Tanabe et al 05]
  - Applied to analysis of XML transducers.

This research:

- Decision Procedure: (sublogics of) alternation-free μ + nominals + backward modalities + functional modalities.
  (variants of enriched μ-calculus)
- Target: analysis of programs that manipulates pointers.

Known results on satisfiability

$\mathcal{L}(\mu_{\mathrm{AF}})$	EXPTIME-complete	[Emerson 90]
$\mathcal{L}(\mu_{\mathrm{AF}}, \mathrm{nom}, \mathrm{back}, \mathrm{func})$	Undecidable	[Bonatti, Peron 04]
$\mathcal{L}(\mu, \mathrm{nom}, \mathrm{back})$	EXPTIME-complete	[Sattler, Vardi 01]
$\mathcal{L}(\mu, \mathrm{nom}, \mathrm{func})$	EXPTIME-complete	[Bonatti et al 06]
$\mathcal{L}(\mu, \mathrm{back}, \mathrm{func})$	EXPTIME-complete	[Bonatti et al 06]
$\mathcal{L}(\mathrm{nom}, \mathrm{back}, \mathrm{func})$	NEXPTIME-complete	[Tobies 00]

- μ: (full) μ-calculus
- μ_{AF}: alternation-free fragment of μ-calculus
- nom: nominals
- back: backward modalities
- func: functional modalities

Our decision procedure

Logic	Time Complexiy
$\mathcal{L}(\mu_{\mathrm{AF}}, \mathrm{nom}, \mathrm{back})$	$2^{\mathcal{O}(n^2 \log n)}$
$\mathcal{L}(\mu_{\mathrm{AF}}, \mathrm{nom}, \mathrm{func})$	$2^{\mathcal{O}(n^2 \log n)}$
$\mathcal{L}(\mu_{\mathrm{AF}}, \mathrm{back}, \mathrm{func})$	$2^{\mathcal{O}(n \log n)}$

- Existing procedures are automata-based.
  - Powerful: no restriction for alternation
  - Complex: $\geq 2^{\mathcal{O}(n^4)}$ even for $\mathcal{L}(\mu)$.
- Ours is by simply enumerating possible nodes.
  - Only for alternation-free fragments.
  - Allows efficient implementation using BDDs.
  - Has successfully been applied to the target.

Syntax

PS $\ni p$	Propositional Symbols
Nom $\ni x$	Nominals
PV $\ni X$	Propositional Variables
FMS $\ni f$	Funcitonal Modality Symbols
GMS $\ni g$	General Modality Symbols
MS $\ni h ::= f \mid g$	Modality Symbols
Mod $\ni m ::= h \mid h^{-1}$	Modalities
Form $\ni \varphi$	Formulas

$$::= p \mid x \mid X \mid \neg\varphi \mid \varphi \vee \varphi \mid \langle m \rangle \varphi \mid \mu X \varphi \mid @x\, \varphi$$

Semantics

Kripke Structure $\mathcal{K} = (S, R, L)$

- S: Set of states
- $R : \mathrm{MS} \to \mathcal{P}(S \times S)$
  - $R(h)$ is the transition relation for modality h.
  - $R(f)$ is a partial function if $f \in \mathrm{FMS}$.
- $L : \mathrm{PS} \cup \mathrm{Nom} \to \mathcal{P}(S)$
  - $\mathcal{K}, s \models p \iff s \in [\![p]\!] \iff s \in L(p)$.
  - $L(x)$ is a singleton $\{L'(x)\}$ if $x \in \mathrm{Nom}$.

$R(m^{-1}) = R(m)^{-1}$

Alternation-freeness

- Alternation-freeness is roughly equal to "μ-ν nest-freeness".
- Precisely, φ is alternation-free if it does *not* have patterns:

$$\varphi = \cdots\cdots \mu X(\cdots\cdots \nu Y(\cdots\cdots X \cdots\cdots)\cdots\cdots)\cdots\cdots$$

or

$$\varphi = \cdots\cdots \nu X(\cdots\cdots \mu Y(\cdots\cdots X \cdots\cdots)\cdots\cdots)\cdots\cdots$$

- $\mathcal{L}(\mu_{\mathrm{AF}})$ allows much simpler decision procedure than $\mathcal{L}(\mu)$.
- It still has sufficient expressive power. Examples:
  - $\mu X(p \vee \langle m \rangle X)$: p is reachable by following m.
  - $\nu Y(q_1 \wedge ([m'](q_2 \wedge [m']Y)))$: q_1 and q_2 holds alternatively along m'-paths.
  - $\mu X(\nu Y(q_1 \wedge ([m'](q_2 \wedge [m']Y))) \vee \langle m \rangle X)$: their combination.
- CTL is a sublogic of $\mathcal{L}(\mu_{\mathrm{AF}})$.

Closure

The closure $\mathrm{cl}(\varphi_{\mathrm{I}})$ of a formula φ_{I} is approximately the set of its subformulas, except for fixed-point operators.

Example: $\varphi_{\mathrm{I}} = \neg p \vee \mu X(p \vee \langle m \rangle X)$

$$\begin{aligned}\mathrm{cl}(\varphi_{\mathrm{I}}) = \{&\varphi_{\mathrm{I}},\\ &\neg p,\ \mu X(p \vee \langle m \rangle X),\\ &p,\qquad p \vee \langle m \rangle \mu X(p \vee \langle m \rangle X),\\ &\qquad\qquad \langle m \rangle \mu X(p \vee \langle m \rangle X)\ \}\end{aligned}$$

Decision procedure for $\mathcal{L}(\mu_{\mathrm{AF}})$

Modification of the decision procedure for CTL [Emerson 90].
A tableau method. Each node is an element of $\mathcal{P}(\mathrm{cl}(\varphi_{\mathrm{I}}))$.

Outline

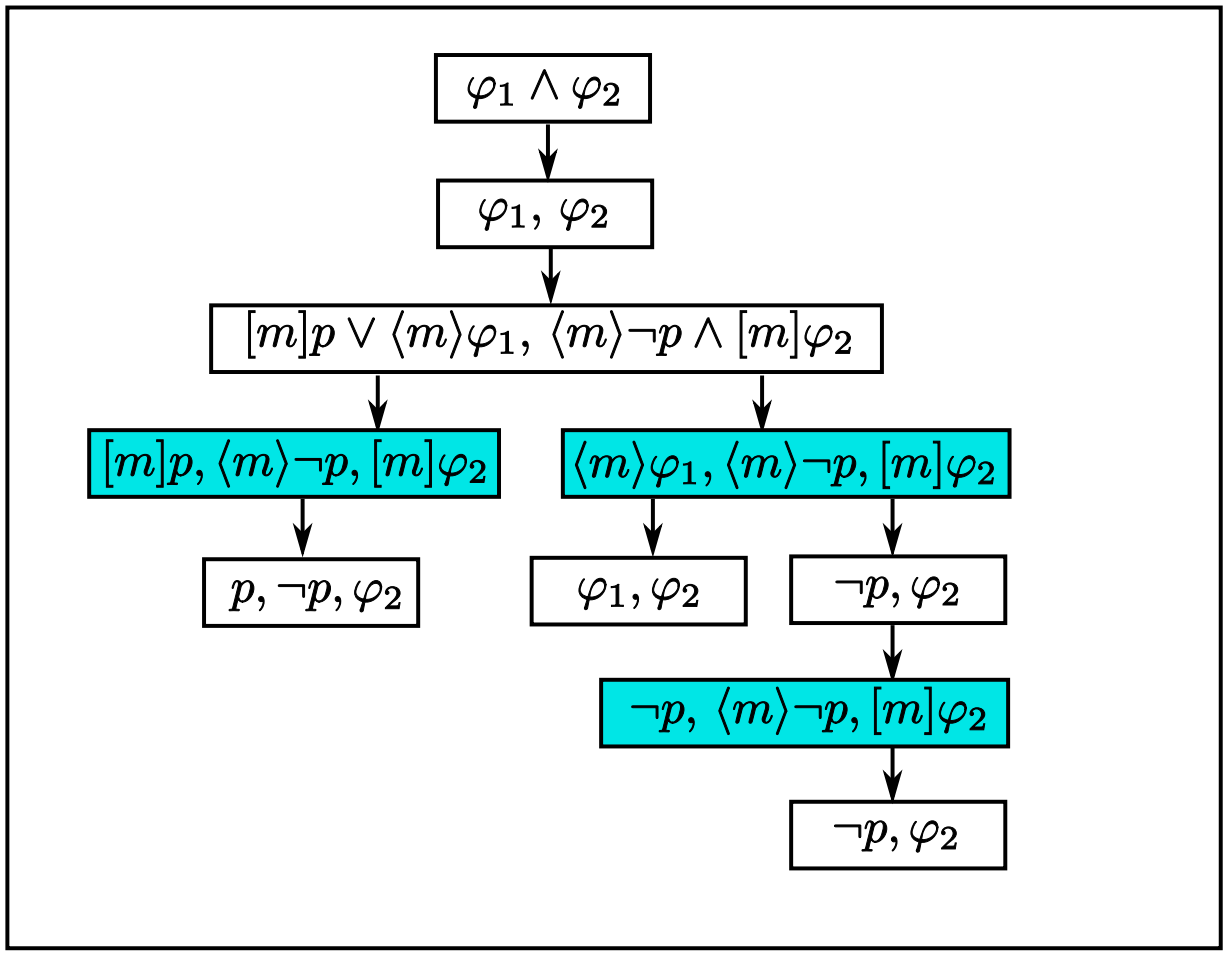
1. Start with initial node $n_0 = \{\varphi_{\mathrm{I}}\}$.
2. Add nodes by decomposing existing nodes.
3. Remove all *inconsistent* nodes.
   - $\neg$-consistency, $\Diamond$-consistency, and μ-consistency.
4. Repeat the previous step until all remaining nodes are consistent.

φ_{I} is satisfiable iff there remains a node that contains φ_{I}.

Decision procedure for $\mathcal{L}(\mu_{\mathrm{AF}})$

$\varphi_{\mathrm{I}} = \varphi_1 \wedge \varphi_2, \qquad \varphi_1 = \mu X([m]p \vee \langle m \rangle X), \quad \varphi_2 = \nu Y(\langle m \rangle \neg p \wedge [m]Y)$

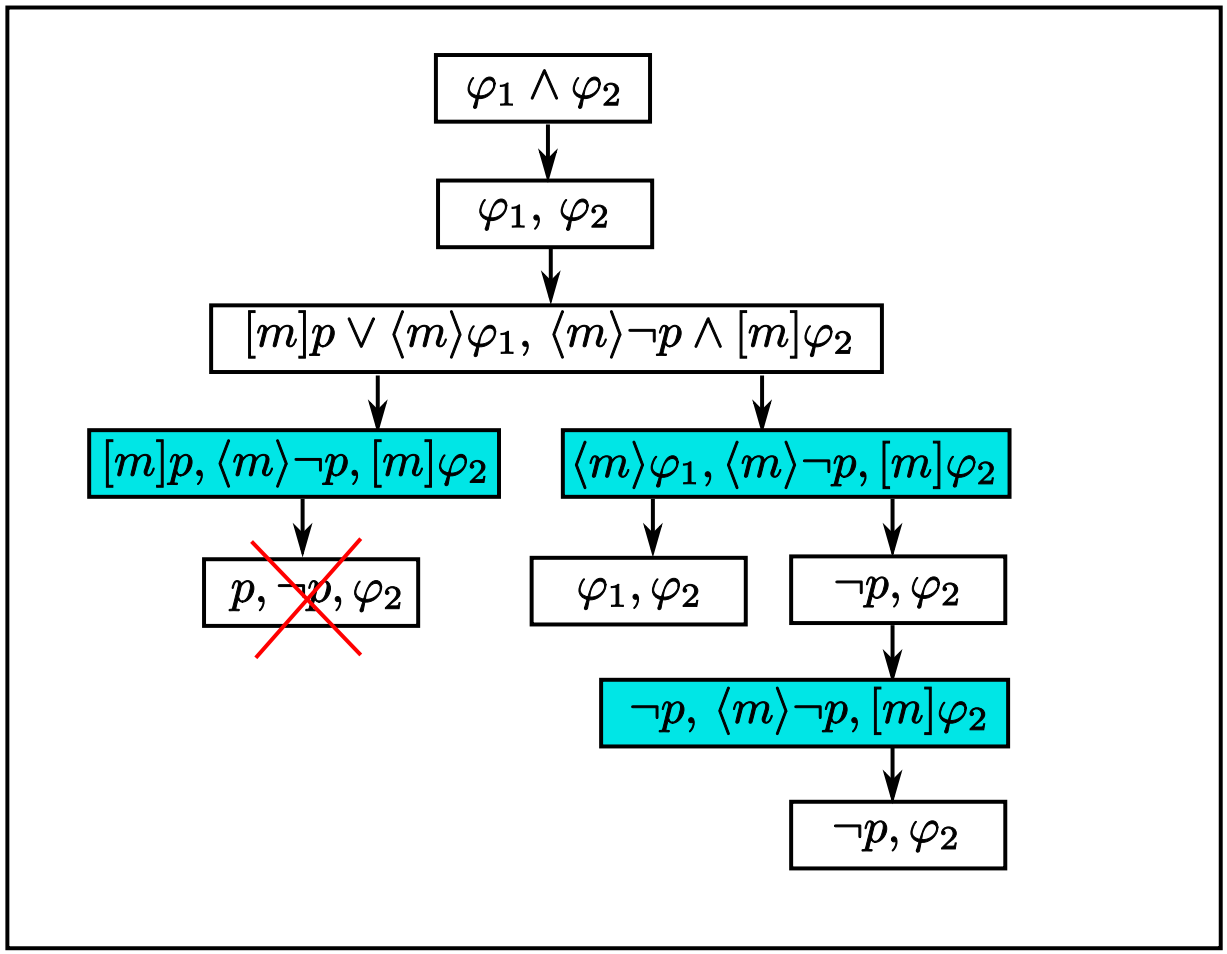
Adding nodes



Decision procedure for $\mathcal{L}(\mu_{\mathrm{AF}})$

$\varphi_{\mathrm{I}} = \varphi_1 \wedge \varphi_2, \qquad \varphi_1 = \mu X([m]p \vee \langle m \rangle X), \quad \varphi_2 = \nu Y(\langle m \rangle \neg p \wedge [m]Y)$

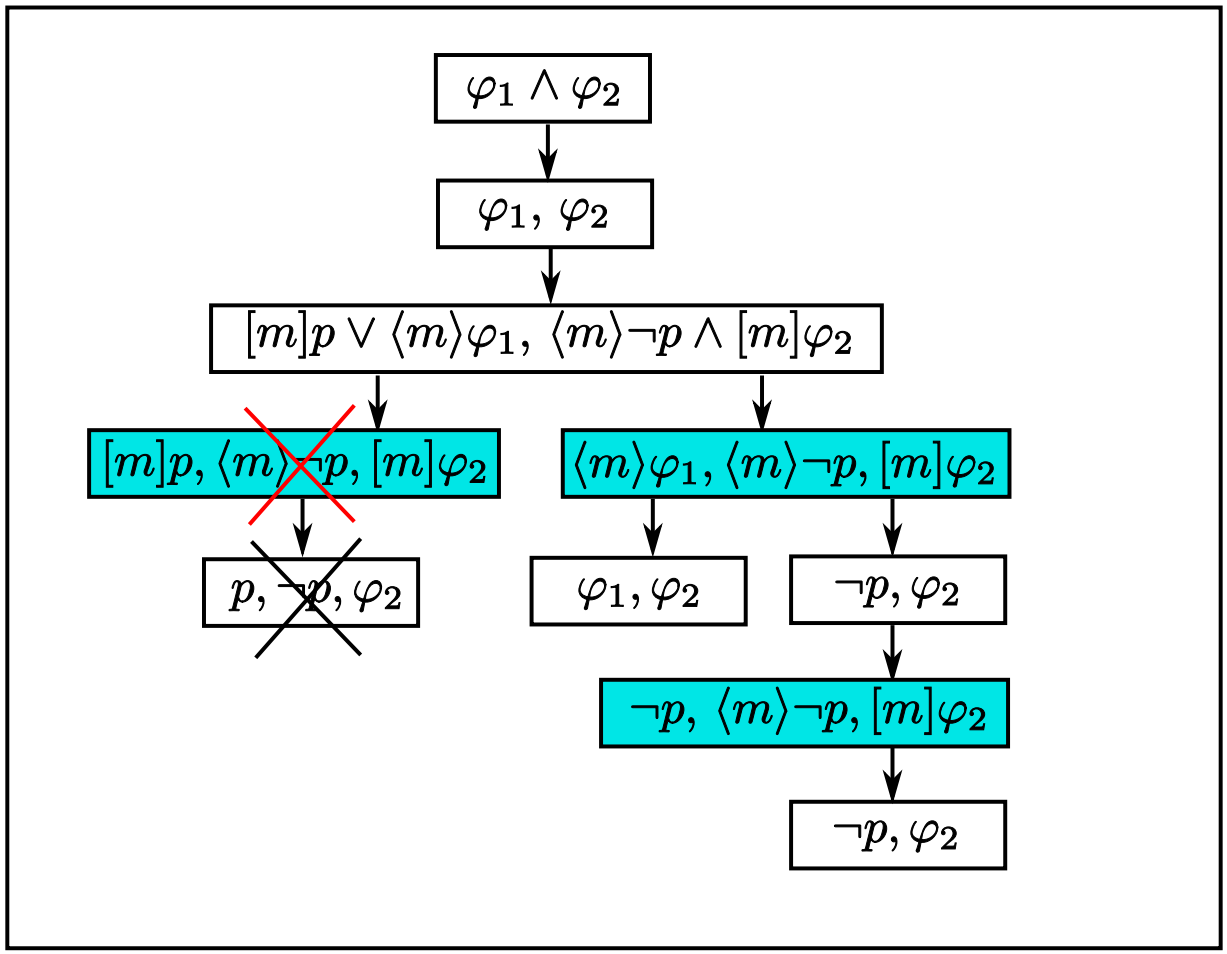
Checking ¬-consistency



Decision procedure for $\mathcal{L}(\mu_{\mathrm{AF}})$

$\varphi_{\mathrm{I}} = \varphi_1 \wedge \varphi_2, \qquad \varphi_1 = \mu X([m]p \vee \langle m\rangle X), \quad \varphi_2 = \nu Y(\langle m\rangle \neg p \wedge [m]Y)$

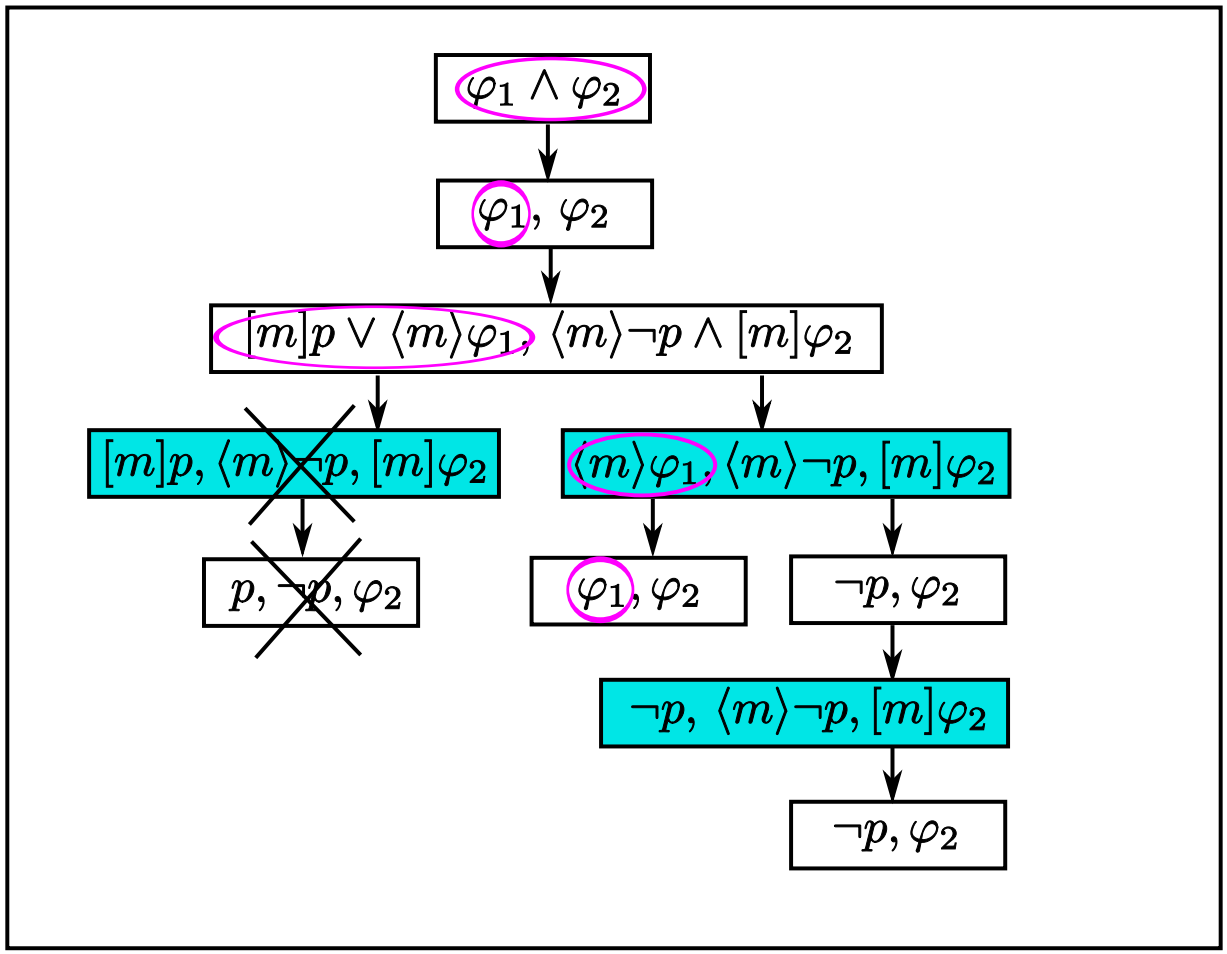
Checking $\Diamond$-consistency



Decision procedure for $\mathcal{L}(\mu_{\mathrm{AF}})$

$\varphi_{\mathrm{I}} = \varphi_1 \wedge \varphi_2, \qquad \varphi_1 = \mu X([m]p \vee \langle m \rangle X), \quad \varphi_2 = \nu Y(\langle m \rangle \neg p \wedge [m]Y)$

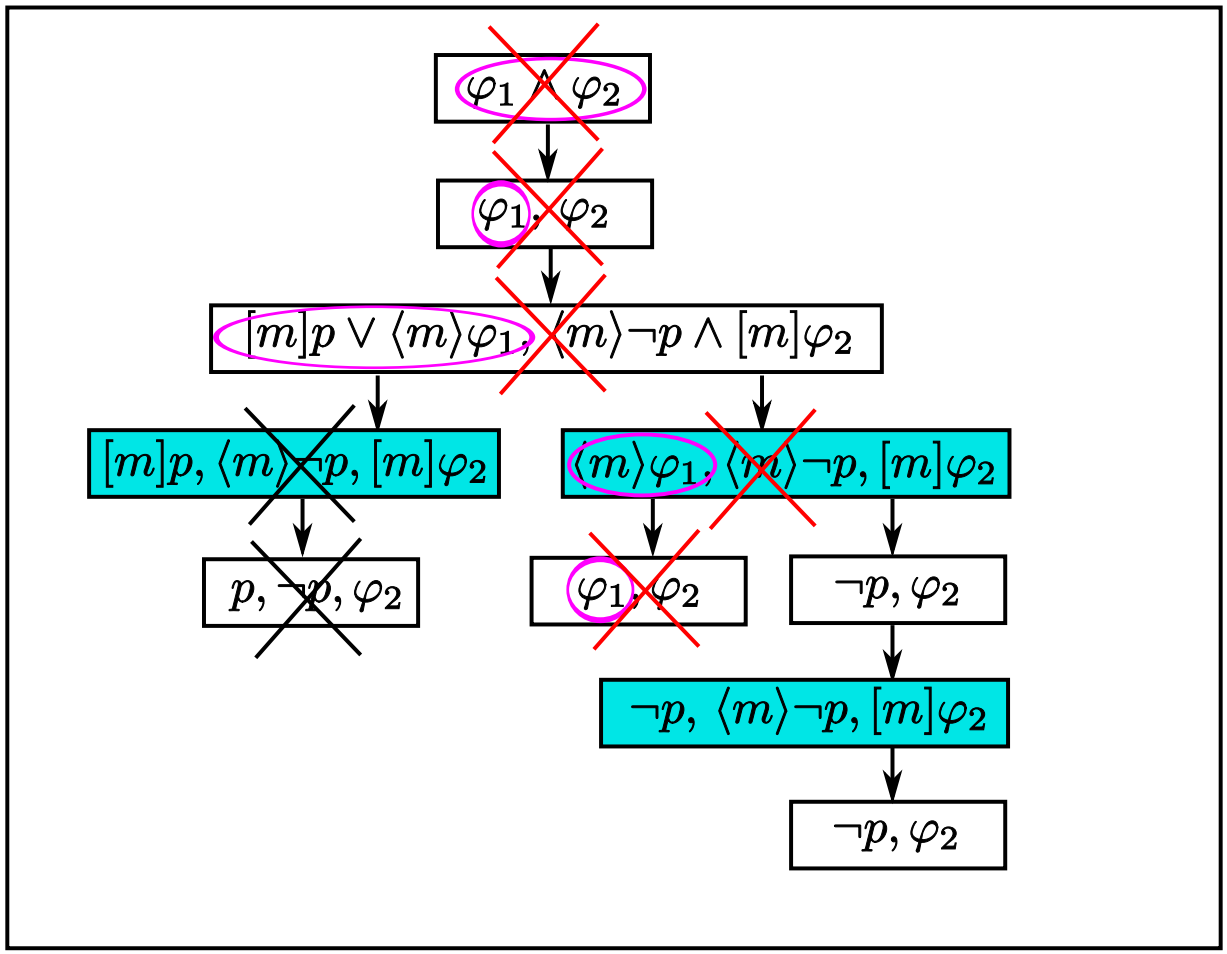
Checking μ-consistency – Remove μ-loops without witnesses



Decision procedure for $\mathcal{L}(\mu_{\mathrm{AF}})$

$\varphi_{\mathrm{I}} = \varphi_1 \wedge \varphi_2, \qquad \varphi_1 = \mu X([m]p \vee \langle m \rangle X), \quad \varphi_2 = \nu Y(\langle m \rangle \neg p \wedge [m] Y)$

Checking μ-consistency – Remove μ-loops without witnesses

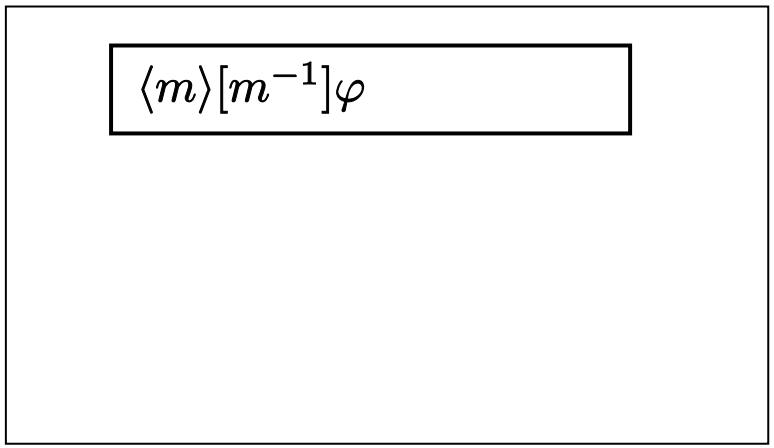


Functional Modality

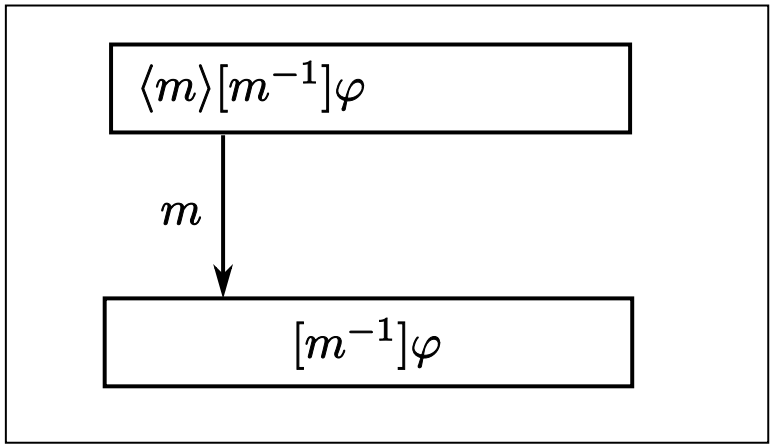
For a node n that contains only atomic formula, $\Diamond$-formula and $\Box$-formula:

- For every formula $\langle g \rangle \varphi$, where $g \in \mathrm{GMS}$, add a node and put formulas φ and all ψ where $[g]\psi \in n$.
- For every $f \in \mathrm{FMS}$, if there is $\langle f \rangle \varphi \in n$, add a node and put all formulas ψ where $\langle f \rangle \psi \in n$ or $[f]\psi \in n$.

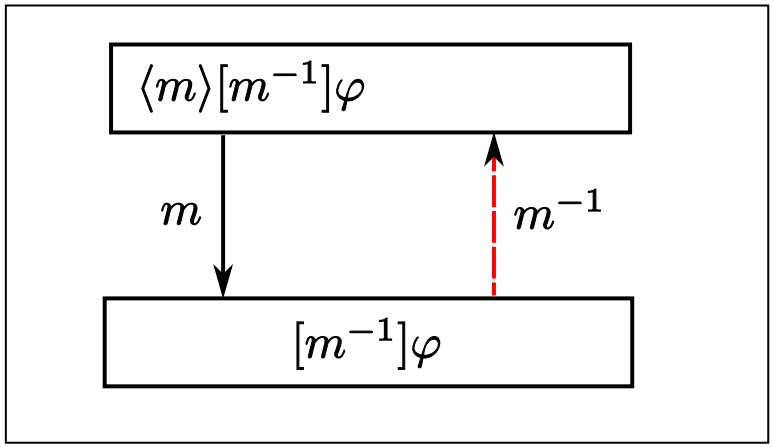
Backward Modality: Problem 1



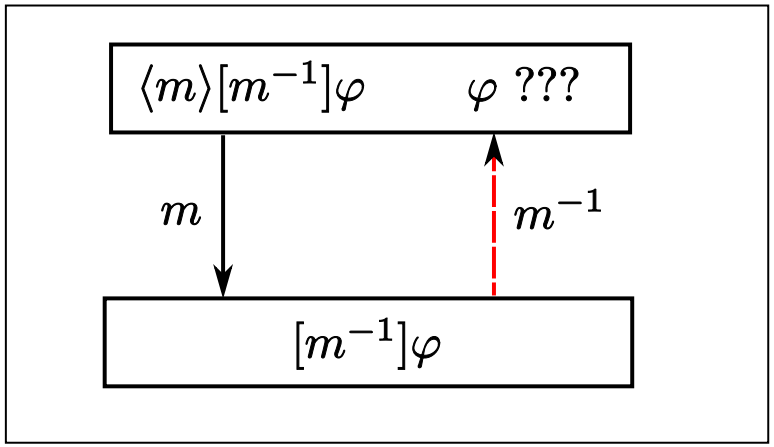
Backward Modality: Problem 1



Backward Modality: Problem 1



Backward Modality: Problem 1



Backward Modality: Solution to Problem 1

Outline of the procedure for given formula φ_{I}:

1. Start with initial node $n = \{\varphi_{\mathrm{I}}\}$.
2. Add nodes by decomposing existing nodes.
3. Remove all *inconsistent* nodes.
   - $\neg$-consistency, $\Diamond$-consistency, and μ-consistency.
4. Repeat the previous step until all remaining nodes are consistent.

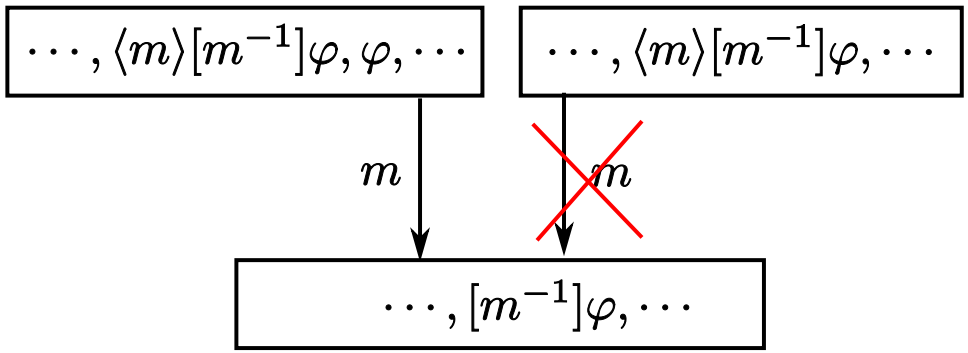
Backward Modality: Solution to Problem 1

Outline of the procedure for given formula φ_{I}:

1. Start with all possible nodes.
2. Remove all *inconsistent* nodes.
   - $\neg$-consistency, $\Diamond$-consistency, and μ-consistency.
3. Repeat the previous step until all remaining nodes are consistent.

Backward Modality: Solution to Problem 1

- all possible nodes = all elements of $\mathcal{P}(\mathrm{cl}(\varphi_{\mathrm{I}}))$
- $(n, n') \in R(m) \iff$
  - $[m]\varphi \in n \implies \varphi \in n'$
  - $[m^{-1}]\varphi \in n' \implies \varphi \in n$

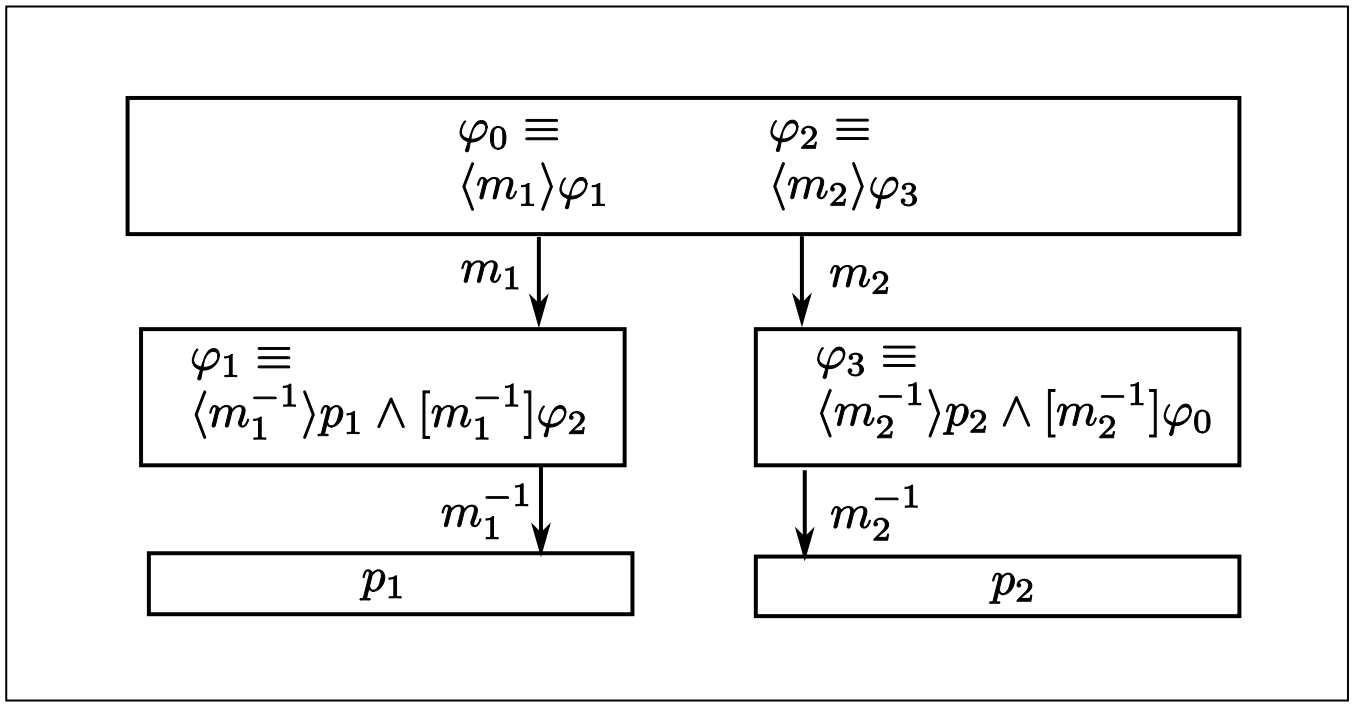


Backward Modality: Problem 2

$\varphi_0 = \mu X(\langle m_1 \rangle(\langle m_1^{-1} \rangle p_1 \wedge [m_1^{-1}](\langle m_2 \rangle(\langle m_2^{-1} \rangle p_2 \wedge [m_2^{-1}](p_2 \vee X))))$

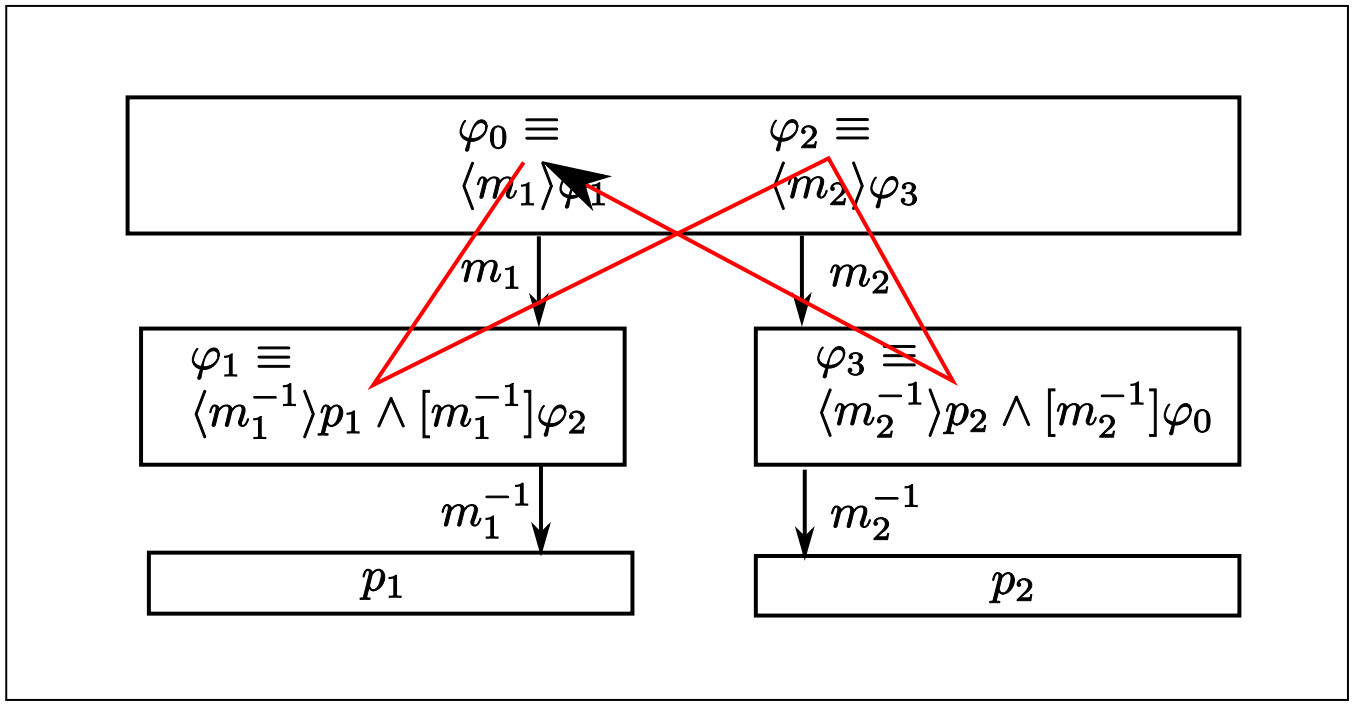
$\varphi_0 \equiv \langle m_1 \rangle \varphi_1, \varphi_1 = \langle m_1^{-1} \rangle p_1 \wedge [m_1^{-1}] \varphi_2$
$\varphi_2 = \langle m_2 \rangle \varphi_3, \varphi_3 = \langle m_2^{-1} \rangle p_2 \wedge [m_2^{-1}] \varphi_0$

$\varphi_I = \varphi_0 \wedge \varphi_2$: unsatisfiable formula

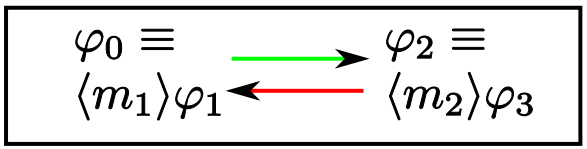


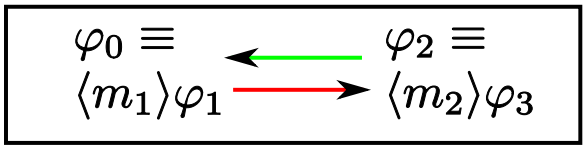
Backward Modality: Problem 2

- All nodes are μ-consistent.
- However, there is a μ-loop without witnesses.



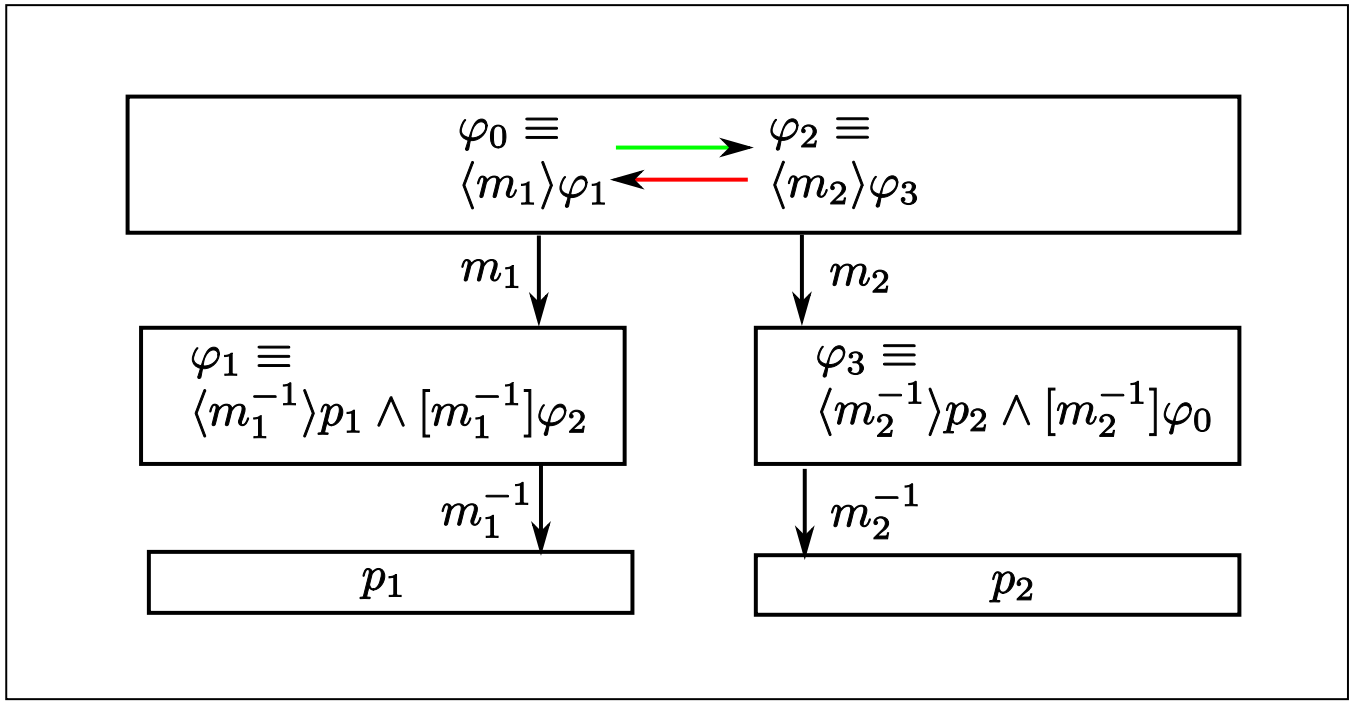
Backward Modality: Solution to Problem 2



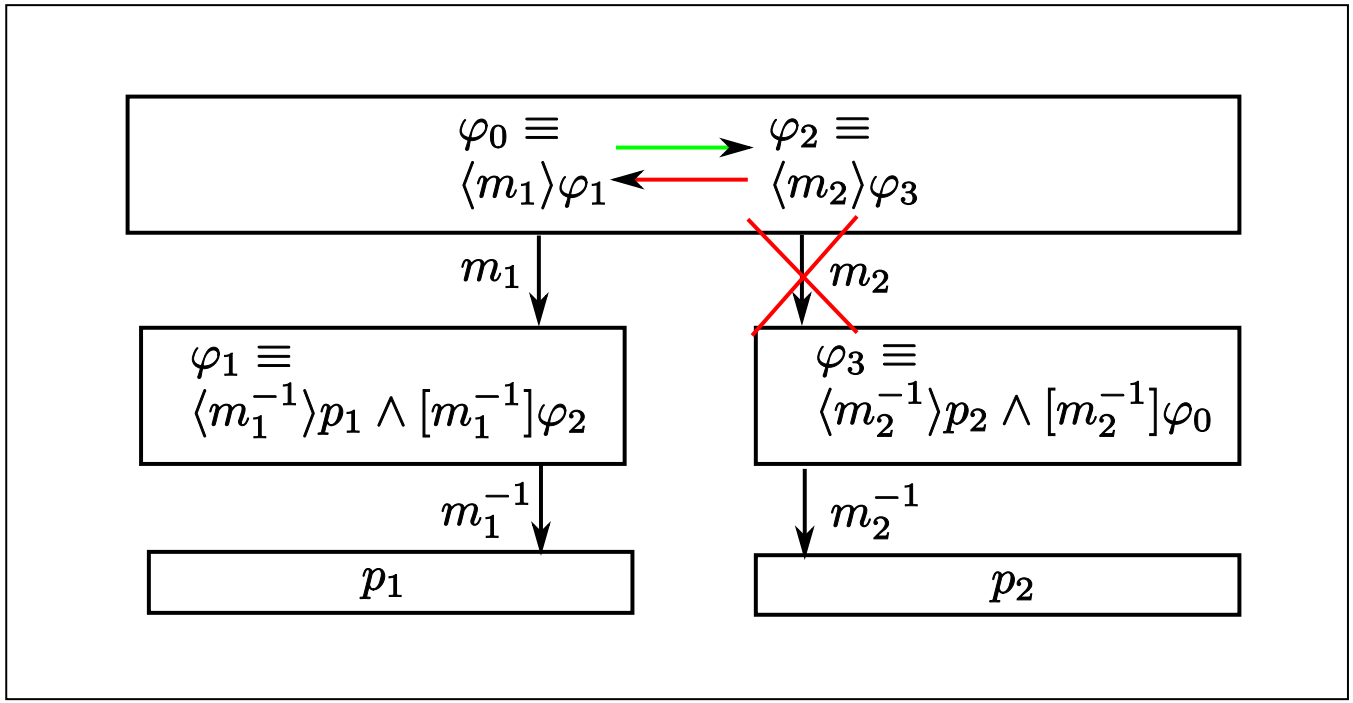


- Each node n is not just a member of $\mathcal{P}(\mathrm{cl}(\varphi_\mathrm{I}))$, but $n = (\Gamma, x)$, where
  - $\Gamma \subseteq \mathrm{cl}(\varphi_\mathrm{I})$
  - x carries information on "expansion direction" (indicated by green and red arrows above) for μ-formulas

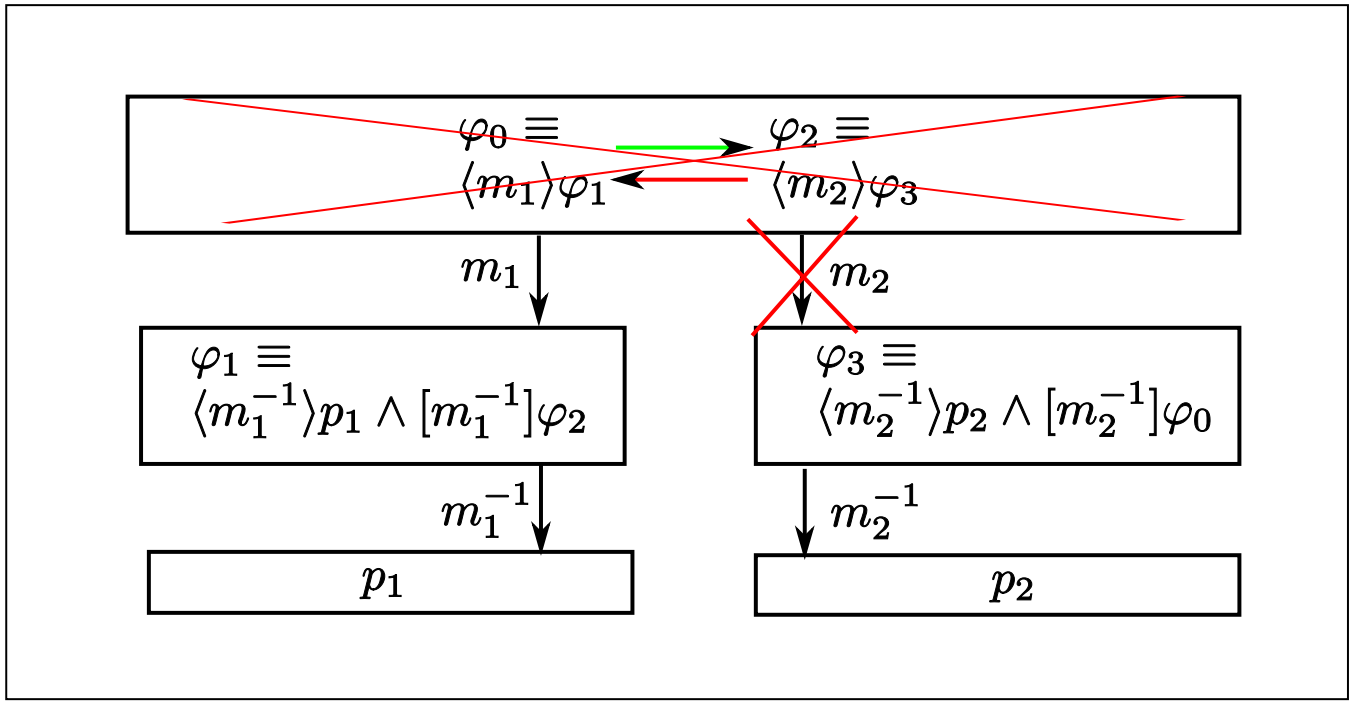
Backward Modality: Solution to Problem 2



Backward Modality: Solution to Problem 2



Backward Modality: Solution to Problem 2



Nominals: pseudo-nominal

If x is nominal,

- $\exists s \in S.\, \mathcal{K}, s \models x$
- $\mathcal{K}, s \models x \;\&\; \mathcal{K}.s' \models x \implies s = s'$.

(Propositional symbol) x is a pseudo-nominal in $\mathcal{K}$, if:

- $\exists s \in S.\, \mathcal{K}, s \models x$
- $\mathcal{K}, s \models x \;\&\; \mathcal{K}.s' \models x \implies \forall \psi \in \mathrm{cl}(\varphi_{\mathrm{I}}).\, \mathcal{K}, s \models \psi \iff \mathcal{K}, s' \models \psi$

Nominals: pseudo-nominal

$g : \mathrm{Nom} \to \mathcal{P}(\mathrm{cl}(\varphi_{\mathrm{I}}))$ is a naming function, if

- $x \in g(x)$
- $x \in g(x') \implies g(x) = g(x')$

Used already in [Sattler, Vardi 01]

If x is a pseudo-nominal in $\mathcal{K}$,

$$g : x \mapsto \{\varphi \in \mathrm{cl}(\varphi_{\mathrm{I}}) \mid L'(x) \models \varphi\}$$

is (well-defined and) a naming function.

Nominals: pseudo-nominal

Outline of the procedure for given formula φ_{I}:

1. Guess a naming function g.
2. Start with all possible nodes.
3. Remove all inconsistent nodes.
   - $\neg$-consistency, $\Diamond$-consistency, and μ-consistency.
   - Node n such that $x \in n$ and $n \neq g(x)$.
4. Repeat the previous step until all remaining nodes are consistent.
5. If there is a node that contains φ_{I}, φ_{I} is satisfiable.
6. Otherwise, Go back to the beginning and try another g. Repeat this process until all g has been checked.

Nominals: duplicated node

- Each x is contained in one node.
- Not necessarily so in the resulting Kripke structure.
  A node may relates to several status.

Nominals: avoiding duplication

Assume $\mathcal{K}, s \models \varphi$ and φ is a μ-formula. Let $\overline{y}(s, \varphi)$ be the number of nominals contained in the witness DAG D for (s, φ).

- If (s', φ') is a successor of (s, φ) in D, then $\overline{y}(s', \varphi') \leq \overline{y}(s, \varphi)$.
- Furthermore, if $s' \models x$ for some $x \in \mathrm{Nom}$, then $\overline{y}(s', \varphi') < \overline{y}(s, \varphi)$.

Now, each node n in the Tab is in the form of $n = (\Gamma, x, y)$.
y carries the information of $\overline{y}$.

Correctness

$$\varphi_{\mathrm{I}} \text{ is satisfiable } \overset{?}{\Longleftrightarrow} \text{ the procedure returns 'yes'}$$

- $\Rightarrow$ holds if φ_{I} is in $\mathcal{L}(\mu_{\mathrm{AF}}, \mathrm{nom}, \mathrm{back}, \mathrm{func})$
- $\Leftarrow$ holds if φ_{I} is in $\mathcal{L}(\mu_{\mathrm{AF}}, \mathrm{back}, \mathrm{func})$, $\mathcal{L}(\mu_{\mathrm{AF}}, \mathrm{nom}, \mathrm{func})$, or $\mathcal{L}(\mu_{\mathrm{AF}}, \mathrm{nom}, \mathrm{back})$,

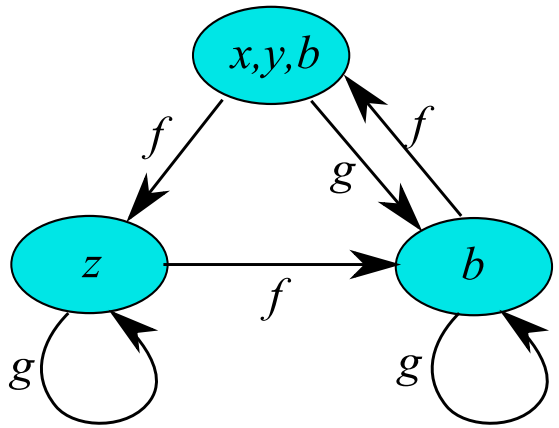
Shape analysis

- Shape analysis: Analysis of programs that manipulates pointers.
- We regard the heap as a Kripke structure.

Example:

```
struct Node {
  Node* f;
  Node* g;
  boolean b;
}

Node* x,y,z;
```



$\mathrm{PS} = \{b\}$ boolean fields

$\mathrm{FMS} = \{f, g\}$ pointer fields

$\mathrm{Nom} = \{x, y, z\}$ pointer type variables

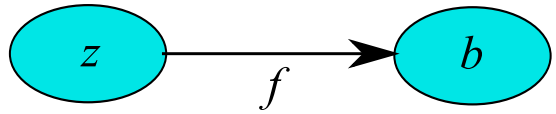
Properties of the heap

Properties of the heap is expressed by formulas.

Example:

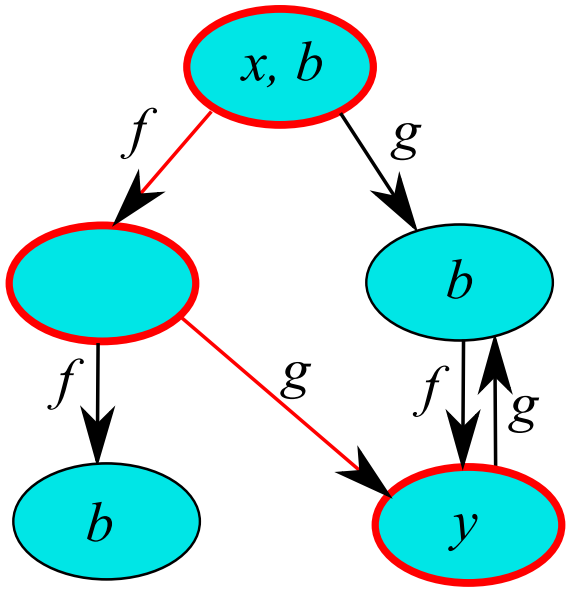
$@z\,\langle f\rangle b$

The f-successor of z satisfies b.



$@x\,\mu X(y \vee (b \wedge \langle f\rangle X) \vee (\neg b \wedge \langle g\rangle X))$

y is reachable from x by following pointer f or g, depending on whether b is satisfied or not.

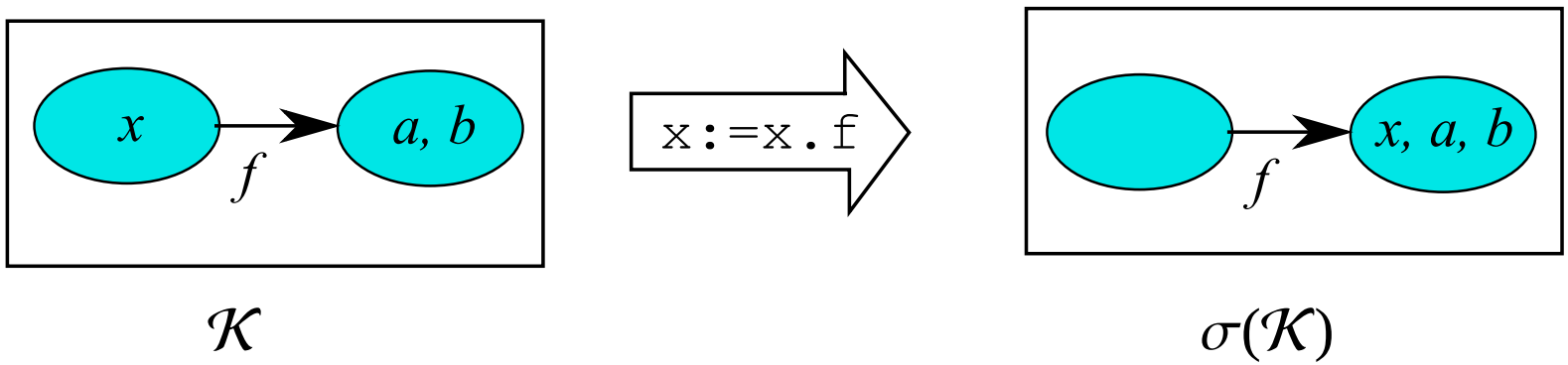


Weakest preconditions

For each “basic operation” σ and formula φ, we can define formula $\mathrm{wp}(\sigma, \varphi)$ such that

$$\mathcal{K} \models \mathrm{wp}(\sigma, \varphi) \iff \sigma(\mathcal{K}) \models \varphi$$

Example: $\sigma =$“`x:=x.f`”, $\varphi = @x\, a \wedge b$



$\mathcal{K}$ $\qquad$ $\sigma(\mathcal{K})$

$$\mathrm{wp}(\sigma, \varphi) = @x\, \langle f \rangle (a \wedge b)$$

Hoare-style proof example

$\{\ @x\,\mathrm{EF}(a)\ \}$

```
y:= nil;
```
$\{\ \psi\ \}$

```
while x != nil do
```
$\{\ \psi \wedge \neg @x\,\mathbf{nil}\ \}$

```
  t := y;
```
$\{\ (@t\,\mathrm{EU}(\neg x, a) \vee @x\,\mathrm{EU}(\neg t, a)) \wedge @x\,\neg\mathbf{nil} \wedge @t\,y\ \}$

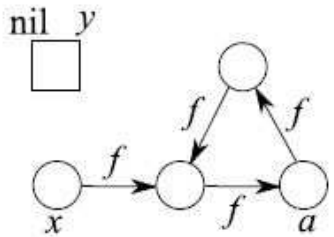
```
  y := x;
```
$\{\ \varphi_1 = (@t\,\mathrm{EU}(\neg y, a) \vee @y\,\mathrm{EU}(\neg t, a)) \wedge @y\,\neg\mathbf{nil} \wedge @y\,x\ \}$

```
  x := y.f;
```
$\{\ \varphi_2 = (@t\,\mathrm{EU}(\neg y, a) \vee @y\,\mathrm{EU}(\neg t, a)) \wedge @y\,\langle f\rangle x\ \}$

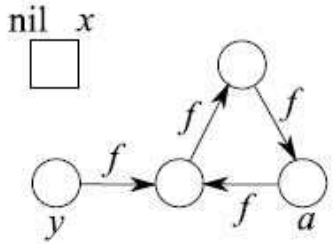
$\{\ \varphi_3 = (@t\,\mathrm{EU}(\neg y, a) \vee @y\,a \vee @x\,\mathrm{EU}(\neg y, a)) \wedge @y\,\langle f\rangle x\ \}$

```
  y.f := t
```
$\{\ \psi\ \}$

```
od
```
$\{\ \psi \wedge @x\,\mathbf{nil}\ \}$ $\{\ @y\,\mathrm{EF}(a)\ \}$



$\mathcal{K}_1$



$\mathcal{K}_2$

Judging Hoare triples

Hoare triple for “basic operation” can be verified using the weakest precondition and a decision procedure

If a decision procedure judges that

$$\varphi \to \mathrm{wp}(\sigma, \psi) \quad \text{is valid,}$$

then hoare triple

$$\{\varphi\}\sigma\{\psi\} \quad \text{is correct.}$$

For this purpose, the decision procedure must be sound, but need not to be complete.

Implementation

A system that helps the user to build Hoare-style proofs is implemented:

- Written in Java.
- Uses BDD (JavaBDD) for set operations.
- Pseudo-nominality.
- $\mathcal{L}(\mu_{\mathrm{AF}}, \mathrm{nom}, \mathrm{func})$ with strongest postconditions.

Performance Examples

id	len	# cl	# nom	time (ms)
regr56	35	11	3	461
listRevNoLeakB	92	25	5	610
listRevSwapA	148	37	6	1199
dswPopC2	188	46	7	2615
dswPushA1	372	54	7	9469

Pentium 4 CPU 2.4GHz, 1GB mem, Windows XP.

Conclusion

Summary

- An implementable decision procedure for variants of enriched modal μ-calculus.
- Application to program analysis.

Future Work

- Implementation of the full decision procedure.
- Decision procedures for other variants, such as:
  - μ-GF, guarded fragments of the first order logic with fixed-point operators.
  - Some variants of multi-valued μ-calculus.

Thank you for your attention.

reg56 =

```
  @ x1 nu Y1 ( !b1 & [n1] Y1 )
& @ x2 nu Y1 ( !b1 & [n1] Y1 )
& @ x1 mu X1 ( <n1> X1 | b1 )
& @ NULL <n1> NULL
```

dswPushA1 =

```
( ( @ p ( p | ( <left> t & !p ) )
    & @ p <left> x
    & @ p mu XC ( ( ( @ t XC & p )
                  | ( !p & <left> XC )
                  | <right> XC
                  | a
                  )
                & ( !marked | p )
                )
    & @ p ( !marked | p )
    & @ p !NULL
    & @ p marked
    & @ x mu X0 ( ( ( ( <right> X0 & swung )
                    | ( !swung
                      & ( ( @ t X0 & p )
                        | ( !p & <left> X0 )
                        )
                      )
                    )
                  & marked
                  & !p
                  )
                | NULL
                )
    & @ a ( ( ( <left> b & !p )
            | ( p & @ t b )
            )
          & <right> c
          & ( !marked | p )
          )
    & @ a !NULL
    & @ a !p
    )
| ( @ p ( p | ( <left> t & !p ) )
    & @ p <left> x
    & @ p ( !marked | p )
    & @ p mu XC ( ( ( @ t XC & p )
                  | ( !p & <left> XC )
                  | <right> XC
                  | a
                  )
                & ( !marked | p )
                )
    & @ x mu X0 ( ( marked
                  & ( ( !swung
                      & ( ( @ t X0 & p )
                        | ( !p & <left> X0 )
                        )
                      & !p
                      )
                    | ( ( swung | p ) & <right> X0 )
                    )
                  & !p
                  )
                | NULL
                )
    & @ p !NULL
    & @ p marked
    & @ a ( ( ( <left> b & !p )
            | ( p & @ t b )
            )
          & <right> c
          & ( !marked | p )
          )
    & @ a !p
    & @ a !NULL
    )
  )
& @ NULL <left> NULL
& @ p !swung
& ( ( ( @ p nu XE ( ( !swung | [right] XE )
                  & [right] nu XC ( ( [right] XC
                                    & [left] XC
                                    & !a
                                    )
```

```
                                    | marked
                                    )
                  & ( [left] XE | swung )
                  )
      | @ a t
      | @ a ( [left] !b
             | marked
             | [right] !c
             )
    )
  & ( @ a ( [left] !b
             | marked
             | [right] !c
             )
    | @ a !t
    )
  & ( @ t nu XC ( ( [right] XC
                    & [left] XC
                    & !a
                    )
                  | marked
                  )
    | @ a t
    | @ a ( [left] !b
             | marked
             | [right] !c
             )
    )
  )
| @ p nu X0 ( ( ( ( !swung | [right] X0 )
                  & ( swung | [left] X0 )
                  )
                | !marked
                )
              & !NULL
              )
| @ a NULL
)
& @ NULL <right> NULL
```