COMPUTER SCIENCE · SAARLAND UNIVERSITY
JUNE 1, 2006

Bachelor's Thesis: Hybrid Logic Revisited

Final Talk by Moritz Hardt

Advisor: Prof. Dr. Gert Smolka
Programming Systems Lab
Winter Term 2005/2006

Overview: Next 30 minutes

1. Motivation
2. Our Approach to Modal Logic
3. Our Decision Procedure
4. Conclusion

Why Modal Logic?

- Many applications in computer science
  - Temporal Logic: Software Verification (A. Pnueli)
  - Description Logic: Artificial Intelligence, Information Retrieval
- Logical interest
  - Model theory, frame definability etc.

Some Modal Formulas

- Propositional Dynamic Logic

$$\langle (x := 8 \mid x := 10) \; ; \; (x := x \text{ mod } 2) \rangle (x = 0)$$

- Linear-Time Temporal Logic

$$\bigcirc(\; \Box(x > 9) \;\wedge\; \Diamond(x = 13) \;)$$

- Hybrid Logic

$$\downarrow x . @u . \Box\Box\Box x$$

Modal Logic Considered *Non-Classical*

- Ad-hoc syntax
  - Nice for applications
- Kripke semantics [Kri63]
  - Meta-level names and quantification

Modal Logic Considered *Non-Classical*

- Ad-hoc syntax
  - Nice for applications
  - Frequently eliminated by standard translations
- Kripke semantics [Kri63]
  - Meta-level names and quantification
  - Partly internalized by standard translations

Modal Logic Considered *Non-Classical*

- Ad-hoc syntax
  - Nice for applications
  - Frequently eliminated by standard translations
- Kripke semantics [Kri63]
  - Meta-level names and quantification
  - Partly internalized by standard translations

comfort of modal reasoning $\leftrightsquigarrow$ **coherence** of classical logic

Modal Logic Considered *Non-Classical*

- Ad-hoc syntax
  - Nice for applications
  - Frequently eliminated by standard translations
- Kripke semantics [Kri63]
  - Meta-level names and quantification
  - Partly internalized by standard translations

comfort of modal reasoning $\leftrightsquigarrow$ **coherence** of classical logic

Trade-off necessary?

Our Logical Base

- Simply-typed lambda calculus
  - Higher-order abstract syntax
  - Standard semantics
- First-order predicate logic
- Equational deduction

Specification	ML
Base Types	B, V
Constants	$0, 1 : \mathsf{B}$
	$\neg : \mathsf{B} \to \mathsf{B}$
	$\wedge, \vee : \mathsf{B} \to \mathsf{B} \to \mathsf{B}$
	$\forall, \exists : (\mathsf{V} \to \mathsf{B}) \to \mathsf{B}$
	$\doteq : \mathsf{V} \to \mathsf{V} \to \mathsf{B}$
	$R : \mathsf{V} \to \mathsf{V} \to \mathsf{B}$
Axioms	See [Smo06]

Propositional variables $f, g : \mathsf{V} \to \mathsf{B}$
Names u, v (either variables $x, y : \mathsf{V}$ or parameters $a, b : \mathsf{V}$)

Modal Operators

Specification includes derived modal operators:

$$\Box x f = \forall y. \neg(Rxy) \vee f y$$

$$\Diamond x f = \exists y. Rxy \wedge f y$$

Modal Operators

Specification includes derived modal operators:

$$\Box x f = \forall y. \neg(Rxy) \vee f y$$

$\Box u(\lambda x.t)$ "At u, **all** direct successors x satisfy t."

$$\Diamond x f = \exists y. Rxy \wedge f y$$

$\Diamond u(\lambda x.t)$ "At u, **some** direct successor x satisfies t."

Traditional Modal Syntax Becomes Notation

- Fix single variable as *point of evaluation*

$$\pi : \mathsf{V}$$

- Specialize variables and operators

$$\mathring{f} \stackrel{\text{def}}{=} f\pi$$

$$\mathring{\Box} t \stackrel{\text{def}}{=} \Box\pi(\lambda\pi.t)$$

We can define traditional modal logics:

$$t, t' \in \mathcal{K} \overset{\text{def}}{=} \mathring{f} \mid \neg t \mid t \wedge t' \mid \mathring{\Box} t$$

Our minimal **modal fragment**:

$$t, t' \in \mathcal{MF} \overset{\text{def}}{=} f u \mid \neg t \mid t \wedge t' \mid \Box u (\lambda x . t)$$

- Certain formulas in $\mathcal{MF}$ do **not** have an equivalent in $\mathcal{K}$

$$fa \wedge \neg(fb)$$

- $\mathcal{MF}$ already provides **naming** and **binding**!

$$\leadsto \text{Hybrid Logic}$$

Hybrid Logic

- Introduces **naming**, **binding** and **identity** to modal logic
- Early work by Arthur Prior in the 1960's
- Modern formulations active research topic in modal logic since the 1990's
  - Areces, Blackburn
  - Horrocks (Description Logic)

- Nominals

$$\mathring{u} \stackrel{\text{def}}{=} \pi \dot{=} u$$

- Satisfaction-Operator

$$@u.t \stackrel{\text{def}}{=} (\lambda\pi.t)u$$

- Down-Operator

$$\downarrow x.t \stackrel{\text{def}}{=} (\lambda x.t)\pi$$

$$t, t' \in \mathcal{HL}(@, \downarrow) \stackrel{\text{def}}{=} \mathring{f} \mid \mathring{u} \mid \neg t \mid t \wedge t' \mid \mathring{\Box} t \mid @u.t \mid \downarrow x.t$$

Our equivalent:

$$t, t' \in \mathcal{MFI} \stackrel{\text{def}}{=} f\,u \mid u \doteq v \mid \neg t \mid t \wedge t' \mid \Box u(\lambda x.t)$$

- Introduces only $\doteq$ to $\mathcal{MF}$
- $\mathcal{HL}(@, \downarrow)$ maps into $\mathcal{MFI}$ via β-reduction
- Inverse mapping [H]

- $\mathcal{HL}(@, \downarrow)$ undecidable [tF05]
- $\mathcal{HL}(@)$ PSPACE-complete [ABM99]

$$t, t' \in \mathcal{HL}(@) \stackrel{\text{def}}{=} \mathring{f} \mid \mathring{a} \mid \neg t \mid t \wedge t' \mid \mathring{\Box} t \mid @a.t$$

- $\mathcal{HL}(@, \downarrow)$ undecidable [tF05]

- $\mathcal{HL}(@)$ PSPACE-complete [ABM99]

$$t, t' \in \mathcal{HL}(@) \stackrel{\text{def}}{=} \mathring{f} \mid \mathring{a} \mid \neg t \mid t \wedge t' \mid \mathring{\Box} t \mid @a.t$$

No $\downarrow$-operator in $\mathcal{MFI}$. How to restrict $\mathcal{MFI}$?

Quasi-Monadicity

Each subterm $u \doteq v$ contains a parameter

- **Quasi-Monadicity**
  Each subterm $u \doteq v$ contains a parameter
- **Monadicity**
  Quasi-M. + Modal operators do not have nested scope

- **Quasi-Monadicity**
  Each subterm $u \doteq v$ contains a parameter
- **Monadicity**
  Quasi-M. + Modal operators do not have nested scope

Not quasi-monadic	$\Diamond a(\lambda x. \Diamond x(\lambda y. y \doteq x))$
Not monadic	$\Box a(\lambda x. \Diamond b(\lambda y. f x))$
Monadic	$\Diamond a(\lambda x. f x \land \Diamond b(\lambda x. f x))$

- **Quasi-Monadicity**
  Each subterm $u \doteq v$ contains a parameter
- **Monadicity**
  Quasi-M. + Modal operators do not have nested scope

Not quasi-monadic	$\Diamond a(\lambda x.\Diamond x(\lambda y.y \doteq x))$
Not monadic	$\Box a(\lambda x.\Diamond b(\lambda y.fx))$
Monadic	$\Diamond a(\lambda x.fx \wedge \Diamond b(\lambda x.fx))$

Prop For each quasi-monadic formula, we can compute an equivalent monadic formula.

Monadic $\mathcal{MFI}$

$$\mathcal{MFI}_1 \stackrel{\text{def}}{=} \{t \in \mathcal{MFI} \mid t \text{ monadic}\}$$

- $\mathcal{HL}(@)$ maps into $\mathcal{MFI}_1$ via β-reduction
- Inverse mapping [H]

Monadic $\mathcal{MFI}$

$$\mathcal{MFI}_1 \stackrel{\text{def}}{=} \{t \in \mathcal{MFI} \mid t \text{ monadic}\}$$

- $\mathcal{HL}(@)$ maps into $\mathcal{MFI}_1$ via β-reduction
- Inverse mapping [H]

Want decision procedure for $\mathcal{MFI}_1$

Decision Procedure

Data structure: **Clause**

finite set of formulas in NNF, interpreted conjunctively

Decision Procedure

Data structure: **Clause**
finite set of formulas in NNF, interpreted conjunctively

Purely monadic clause closed monadic formulas

Monadic clause monadic formulas plus "edges" Ruv

Decision Procedure

Data structure: **Clause**
finite set of formulas in NNF, interpreted conjunctively

Purely monadic clause closed monadic formulas

Monadic clause monadic formulas plus "edges" Ruv

Is a purely monadic clause satisfiable?

Decision Procedure

Data structure: **Clause**
finite set of formulas in NNF, interpreted conjunctively

Purely monadic clause closed monadic formulas

Monadic clause monadic formulas plus “edges” Ruv

Is a purely monadic clause satisfiable?

Find out by **saturation**: $C \to C \cup \{s\}$
Meaningful information s inferred from C

Design Space: Saturation *Conditions*

When is a clause C **saturated**?

Design Space: Saturation *Conditions*

When is a clause C **saturated**?

$(\mathcal{S}_c)$ C is not trivial (no $t, \neg t$ or $\neg(t \doteq t)$ in C)

Design Space: Saturation *Conditions*

When is a clause C **saturated**?

($\mathcal{S}_c$) C is not trivial (no $t, \neg t$ or $\neg(t \doteq t)$ in C)

($\mathcal{S}_\wedge$) If $s \wedge t \in C$, then $\{s, t\} \subseteq C$.

($\mathcal{S}_\vee$) If $s \vee t \in C$, then $s \in C$ or $t \in C$.

Design Space: Saturation *Conditions*

When is a clause C **saturated**?

($\mathcal{S}_c$) C is not trivial (no $t, \neg t$ or $\neg(t \doteq t)$ in C)

($\mathcal{S}_\wedge$) If $s \wedge t \in C$, then $\{s, t\} \subseteq C$.

($\mathcal{S}_\vee$) If $s \vee t \in C$, then $s \in C$ or $t \in C$.

($\mathcal{S}_\Diamond$) If $\Diamond ut \in C$, then $\{Rux, t{\downarrow}x\} \subseteq C$ for some x.

Design Space: Saturation *Conditions*

When is a clause C **saturated**?

($\mathcal{S}_c$) C is not trivial (no $t, \neg t$ or $\neg(t \doteq t)$ in C)

($\mathcal{S}_\wedge$) If $s \wedge t \in C$, then $\{s, t\} \subseteq C$.

($\mathcal{S}_\vee$) If $s \vee t \in C$, then $s \in C$ or $t \in C$.

($\mathcal{S}_\Diamond$) If $\Diamond ut \in C$, then $\{Rux, t{\downarrow}x\} \subseteq C$ for some x.

($\mathcal{S}_\Box$) If $\Box ut \in C$ and $Ruv \in C$, then $t{\downarrow}v \in C$.

Design Space: Saturation *Conditions*

When is a clause C **saturated**?

($\mathcal{S}_c$) C is not trivial (no $t, \neg t$ or $\neg(t \doteq t)$ in C)

($\mathcal{S}_\wedge$) If $s \wedge t \in C$, then $\{s, t\} \subseteq C$.

($\mathcal{S}_\vee$) If $s \vee t \in C$, then $s \in C$ or $t \in C$.

($\mathcal{S}_\Diamond$) If $\Diamond ut \in C$, then $\{Rux, t\!\downarrow\!x\} \subseteq C$ for some x.

($\mathcal{S}_\Box$) If $\Box ut \in C$ and $Ruv \in C$, then $t\!\downarrow\!v \in C$.

($\mathcal{S}_{\doteq}^s$) If $u \doteq v \in C$, then $v \doteq u \in C$.

($\mathcal{S}_{\doteq}$) If $u \doteq a \in C$ and $t \in C$, then $t[u := a] \in C$.

Model Existence

Thm Saturated **monadic** clauses are satisfiable.

Model Existence

Thm Saturated **monadic** clauses are satisfiable.

- Given saturated clause, construct a satisfying interpretation
- Difficulty: Weak identity conditions!

Computational Counterpart: Saturation *Rules*

($\mathcal{C}_\wedge$) If $s \wedge t \in C$, add s and t.

($\mathcal{C}_\vee$) If $s \vee t \in C$ and $s \notin C, t \notin C$, add s or t.

Computational Counterpart: Saturation *Rules*

($\mathcal{C}_\wedge$) If $s \wedge t \in C$, add s and t.

($\mathcal{C}_\vee$) If $s \vee t \in C$ and $s \notin C, t \notin C$, add s or t.

($\mathcal{C}_\Diamond$) If $\Diamond ut \in C$ and $\Diamond ut$ not expanded in C, add Rux and $t\!\downarrow\!x$ for fresh x.

Computational Counterpart: Saturation *Rules*

($\mathcal{C}_\wedge$) If $s \wedge t \in C$, add s and t.

($\mathcal{C}_\vee$) If $s \vee t \in C$ and $s \notin C, t \notin C$, add s or t.

($\mathcal{C}_\Diamond$) If $\Diamond ut \in C$ and $\Diamond ut$ not expanded in C, add Rux and $t{\downarrow}x$ for fresh x.

($\mathcal{C}_\Box$) If $\Box ut \in C$ and $Ruv \in C$, add $t{\downarrow}v$.

Computational Counterpart: Saturation *Rules*

($\mathcal{C}_\wedge$) If $s \wedge t \in C$, add s and t.

($\mathcal{C}_\vee$) If $s \vee t \in C$ and $s \notin C, t \notin C$, add s or t.

($\mathcal{C}_\Diamond$) If $\Diamond ut \in C$ and $\Diamond ut$ not expanded in C, add Rux and $t{\downarrow}x$ for fresh x.

($\mathcal{C}_\Box$) If $\Box ut \in C$ and $Ruv \in C$, add $t{\downarrow}v$.

($\mathcal{C}^s_{\doteq}$) If $u \doteq v \in C$, add $v \doteq u$.

($\mathcal{C}_{\doteq}$) If $u \doteq a \in C$ and $t \in C$, add $t[u := a]$.

Computational Counterpart: Saturation *Rules*

($\mathcal{C}_\wedge$) If $s \wedge t \in C$, add s and t.

($\mathcal{C}_\vee$) If $s \vee t \in C$ and $s \notin C, t \notin C$, add s or t.

($\mathcal{C}_\Diamond$) If $\Diamond ut \in C$ and $\Diamond ut$ not expanded in C, add Rux and $t{\downarrow}x$ for fresh x.

($\mathcal{C}_\Box$) If $\Box ut \in C$ and $Ruv \in C$, add $t{\downarrow}v$.

($\mathcal{C}^s_{\doteq}$) If $u \doteq v \in C$, add $v \doteq u$.

($\mathcal{C}_{\doteq}$) If $u \doteq a \in C$ and $t \in C$, add $t[u := a]$.

$C \to D$ iff $C \subset D$ and D is obtained from C by applying one saturation rule

Computational Counterpart: Saturation *Rules*

($\mathcal{C}_\wedge$) If $s \wedge t \in C$, add s and t.

($\mathcal{C}_\vee$) If $s \vee t \in C$ and $s \notin C, t \notin C$, add s or t.

($\mathcal{C}_\Diamond$) If $\Diamond ut \in C$ and $\Diamond ut$ not expanded in C, add Rux and $t{\downarrow}x$ for fresh x.

($\mathcal{C}_\Box$) If $\Box ut \in C$ and $Ruv \in C$, add $t{\downarrow}v$.

($\mathcal{C}^s_{\doteq}$) If $u \doteq v \in C$, add $v \doteq u$.

($\mathcal{C}_{\doteq}$) If $u \doteq a \in C$ and $t \in C$, add $t[u := a]$.

$C \to D$ iff $C \subset D$ and D is obtained from C by applying one saturation rule

$C \to D_1, D_2$ **don't know** if applied ($\mathcal{C}_\vee$),

$C \to D$ **don't care** otherwise

Wanted: Key Properties

Soundness If $C \rightarrow D$ **don't care**, then C is satisfiable if and only if D is satisfiable.
If $C \rightarrow D_1, D_2$ **don't know**, then C is satisfiable if and only if D_1 is satisfiable or D_2 is satisfiable.

Wanted: Key Properties

Soundness If $C \to D$ **don't care**, then C is satisfiable if and only if D is satisfiable.
If $C \to D_1, D_2$ **don't know**, then C is satisfiable if and only if D_1 is satisfiable or D_2 is satisfiable.

Completeness If C cannot be extended by a saturation rule, then C is satisfiable *iff* it is not trivial.

Wanted: Key Properties

Soundness If $C \to D$ **don't care**, then C is satisfiable if and only if D is satisfiable.
If $C \to D_1, D_2$ **don't know**, then C is satisfiable if and only if D_1 is satisfiable or D_2 is satisfiable.

Completeness If C cannot be extended by a saturation rule, then C is satisfiable *iff* it is not trivial.

Termination No infinite path $C_0 \to C_1 \to C_2 \to \ldots$

Checking Key Properties

Soundness Simple.

Checking Key Properties

Soundness Simple.

Completeness Suppose clause cannot be extended by a rule!
If trivial, then not satisfiable
If not trivial, then saturated, thus satisfiable by Model Existence

Checking Key Properties

Soundness Simple.

Completeness Suppose clause cannot be extended by a rule!
If trivial, then not satisfiable
If not trivial, then saturated, thus satisfiable by Model Existence

Termination ?

Checking Key Properties

Soundness Simple.

Completeness Suppose clause cannot be extended by a rule!
If trivial, then not satisfiable
If not trivial, then saturated, thus satisfiable by Model Existence

Termination ?

- ✐ Saturation increases clause size, how to obtain a bound?

Checking Key Properties

Soundness Simple.

Completeness Suppose clause cannot be extended by a rule!
If trivial, then not satisfiable
If not trivial, then saturated, thus satisfiable by Model Existence

Termination ?

- Saturation increases clause size, how to obtain a bound?
- Bound the number of variables introduced by $(\mathcal{C}_\Diamond)$

Termination: Observation

Partition clause C (excluding edges) into

$C_a = \{t \in C \mid t \text{ closed}\}$

$C_x = \{t \in C \mid \mathsf{FV}t = \{x\}\}$ for each $x \in \mathsf{FV}C$

Termination: Observation

Partition clause C (excluding edges) into
$C_a = \{t \in C \mid t \text{ closed}\}$
$C_x = \{t \in C \mid \mathsf{FV}t = \{x\}\}$ for each $x \in \mathsf{FV}C$

- Degree $\mathsf{deg}C = \mathsf{max}_{t \in C} |t|$ does not increase

Termination: Observation

Partition clause C (excluding edges) into
$C_a = \{t \in C \mid t \text{ closed}\}$
$C_x = \{t \in C \mid \mathsf{FV}t = \{x\}\}$ for each $x \in \mathsf{FV}C$

- Degree $\deg C = \max_{t \in C} |t|$ does not increase
- When applying $(\mathcal{C}_\Diamond)$, resulting in a new variable x
  - We have a **unique** term $\Diamond ut$ that "justifies" x
  - We receive witness $\{Rux, t\!\downarrow\! x\}$

Termination: Observation

Partition clause C (excluding edges) into
$C_a = \{t \in C \mid t \text{ closed}\}$
$C_x = \{t \in C \mid \mathsf{FV}t = \{x\}\}$ for each $x \in \mathsf{FV}C$

- Degree $\mathsf{deg}C = \max_{t \in C} |t|$ does not increase
- When applying $(\mathcal{C}_\Diamond)$, resulting in a new variable x
  - We have a **unique** term $\Diamond ut$ that “justifies” x
  - We receive witness $\{Rux, t\downarrow x\}$
  - If $t \in C_x$, then there is $s \in C_u$ with $|t| < |s|$

Central Invariants: *Admissibility*

C is n-**admissible** if

1. C is monadic and $\deg C \leq n$.

Central Invariants: *Admissibility*

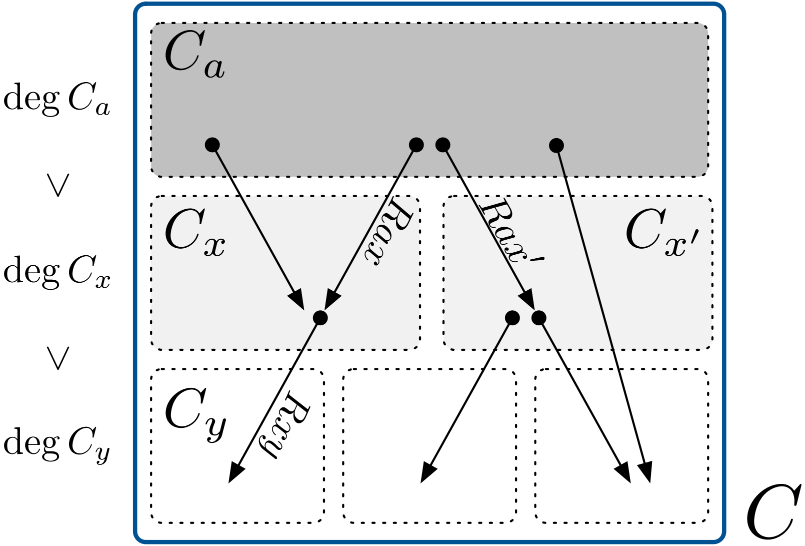
C is n-**admissible** if

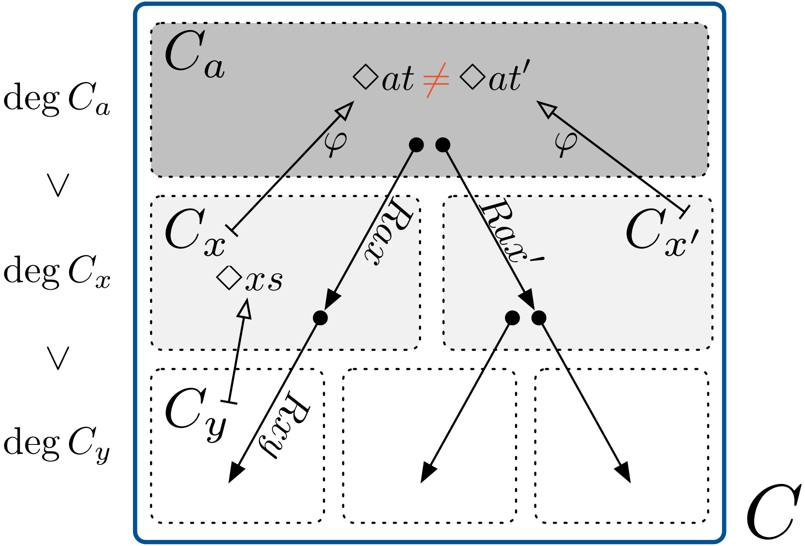
1. C is monadic and deg$C \leq n$.
2. If $Rux \in C$, then deg$C_u >$ degC_x.

Central Invariants: *Admissibility*

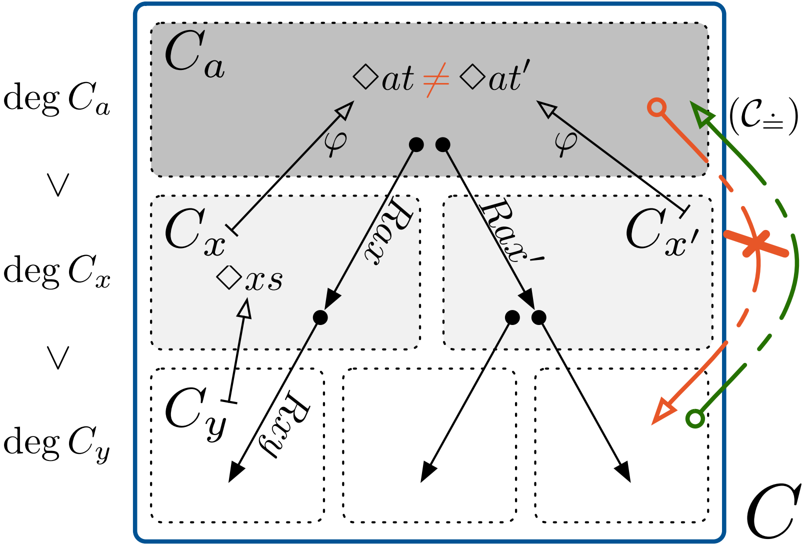
C is n-**admissible** if there is $\varphi \in \mathsf{FV}C \to C$ such that

1. C is monadic and $\mathsf{deg}C \leq n$.
2. If $Rux \in C$, then $\mathsf{deg}C_u > \mathsf{deg}C_x$.
3. φ is injective and
   $\forall x \in \mathsf{FV}C : \exists u, t : \varphi x = \Diamond ut \wedge \{Rux, t{\downarrow}x\} \subseteq C$


$\deg C_a$
$\vee$
$\deg C_x$
$\vee$
$\deg C_y$
C_a
C_x
$C_{x'}$
C_y
Rax
Rax'
Rxy
C



$\deg C_a$
$\vee$
$\deg C_x$
$\vee$
$\deg C_y$
C_a
$\Diamond at \neq \Diamond at'$
φ
φ
C_x
$\Diamond xs$
Rax
Rax'
$C_{x'}$
C_y
Rxy
C


$\deg C_a$
$\vee$
$\deg C_x$
$\vee$
$\deg C_y$
C_a
$\Diamond at \neq \Diamond at'$
φ
φ
$(\mathcal{C}_{\doteq})$
C_x
$\Diamond xs$
Rax
Rax'
$C_{x'}$
C_y
Rxy
C

- Prove n-admissibility is preserved by saturation
- Obtain exponential bound on the size of n-admissible clauses
- Purely monadic clause C is deg$C$-admissible

Thm By means of saturation we can decide whether or not a purely monadic clause is satisfiable.

Conclusion: Modal Logic

- Traditional modal syntax & semantics we still consider harmful
- First-order predicate logic as such syntactically too weak to cope with modal logic

Conclusion: Modal Logic

- Traditional modal syntax & semantics we still consider harmful
- First-order predicate logic as such syntactically too weak to cope with modal logic

Higher-order syntax + First-order predicate logic

Conclusion: Decision Procedure

- Local termination arguments & no external data structures as opposed to [Tza99, BB05].
- Fully internal deduction as in [Bla00], but still explicit access relation
- Fewer and simpler rules for identities

Conclusion: Decision Procedure

- Local termination arguments & no external data structures as opposed to [Tza99, BB05].
- Fully internal deduction as in [Bla00], but still explicit access relation
- Fewer and simpler rules for identities

Our approach: Appropriate for decision procedures!

Further Work

- Generalizing modal logics in our system, e.g., complex relational argument (subsumes universal modalities):

$$\Diamond(\lambda x \lambda y.1)af$$

- Space optimal saturation algorithm for $\mathcal{MFI}_1$
  - Previous results not saturation-based
  - Our PSPACE saturation algorithm submitted to HyLo 2006

- More about decision procedures

Thank you for your attention! 

C. Areces, P. Blackburn, and M. Marx.
A road-map on complexity for hybrid logics.
In J. Flum and M. Rodríguez-Artalejo, editors, *Computer Science Logic*, number 1683 in LNCS, pages 307–321. Springer, 1999.

Thomas Bolander and Torben Bräuner.
Two Tableau-Based Decision Procedures for Hybrid Logic, volume 194 of *Informatik-Berichte*.
4th Workshop: Methods for Modalities, Proceedings, 2005.

Patrick Blackburn.
Internalizing labelled deduction.
Journal of Logic and Computation, 10(1):137 – 168, 2000.

Moritz Hardt.
Bachelor's Thesis: Hybrid Logic Revisited.
Saarland University, http://www.ps.uni-sb.de/∼hardt/hlrev.html, 2006.

Saul A. Kripke.
Semantical analysis of modal logic I: Normal modal propositional calculi.
Zeitschrift für Mathematische Logik und Grundlagen der Mathematik, 9:67–96, 1963.

Gert Smolka.
Lecture Notes: Introduction to Computational Logic.
Saarland University, http://www.ps.uni-sb.de/courses/cl-ss06/script/index.html, 2006.

Balder ten Cate and Massimo Franceschet.
On the complexity of hybrid logics with binders, 2005.

M. Tzakova.
Tableau calculi for hybrid logics.
In N. V. Murray, editor, *Analytic Tableaux and Related Methods, TABLEAUX'99*, volume 1617 of *LNAI*. Springer, 1999.