Incremental Decision Procedures for Modal Logic with Eventualities and Nominals

Gert Smolka

Saarland University

DL 2011
Barcelona, July 15, 2011

My First Encounter with DL

- 1984 Brachman and Levesque (AAAI, Austin)
  - $C, D ::= A \mid C \sqcap D \mid \forall R.C \mid \exists R.C$
  - Concepts describe sets of individuals
  - Roles are binary relations on individuals
  - Notation for fragment of FOL
  - Subsumption coNP-hard (but $O(n^2)$ if $\exists R.\top$)
- 1988 Schmidt-Schauß and Smolka (AI Journal, 1991)
  - Closure under complement, $\mathcal{ALC}$
  - $C, D ::= \ A \mid \neg C \mid C \sqcap D \mid C \sqcup D \mid \forall R.C \mid \exists R.C$
  - Subsumption $\cong$ satisfiability, PSPACE-complete
  - New decision method, evolved in tableau-based method
- 1991 Klaus Schild (IJCAI)
  - $\mathcal{ALC} \cong$ modal logic K
  - PSPACE-completeness first shown by Ladner 1977

Problem Considered in This Talk

- $\mathcal{ALC}$ plus nominals plus R^* (reflexive transitive closure)

$$\begin{aligned} s,t ::= \;& p \mid s \wedge t \mid \forall R.s \mid \forall R^*.s \\ \mid \;& \neg p \mid s \vee t \mid \exists R.s \mid \exists R^*.s \end{aligned}$$

- Eventualities are formulas of the form $\exists R^*.s$
- Nominals are p's that hold for exactly one individual
- Incremental decision procedures for satisfiability
- Challenge comes from combination of eventualities and nominals

Acknowledgement: Joint work with Mark Kaminski

- Terminating Tableaux for Hybrid Logic with Eventualities
  IJCAR 2010, Edinburgh
- Clausal tableaux for hybrid PDL
  Tech. Report 2010
- Clausal Graph Tableaux for Hybrid Logic with Eventualities and Difference
  LPAR 2010, Yogyakarta
- Correctness and Worst-case Optimality of Pratt-style Decision Procedures for Modal and Hybrid Logics
  Tableaux 2011, Bern (with Thomas Schneider)
- Correctness of an Incremental and Worst-case Optimal Decision Procedure for Modal Logic with Eventualities
  Tech. Report 2011

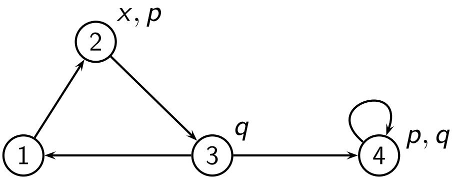
Box and Diamond Notation

- We restrict to a single atomic role R
  (extension to multiple atomic roles is straightforward)
- We use box and diamond notation

$$
\begin{aligned}
\Box s &:= \forall R.s \\
\Box^* s &:= \forall R^*.s \\
\Box^+ s &:= \forall R.\forall R^*.s
\end{aligned}
$$

$$
\begin{aligned}
\Diamond s &:= \exists R.s \\
\Diamond^* s &:= \exists R^*.s \\
\Diamond^+ s &:= \exists R.\exists R^*.s
\end{aligned}
$$

We Think of Models as Transition Systems



- Individuals appear as states
- Atomic concepts and nominals label states
- Nominals label exactly one state
- $\Box s$ holds at a state w if all successors of w satisfy s
- $\Box^* s$ holds at a state w if all states reachable from w satisfy s
- $\Diamond s$ holds at a state w if some successor of w satisfy s
- $\Diamond^* s$ holds at a state w if some state reachable from w satisfy s

Equivalences

$$\Box^* s \equiv s \wedge \Box^+ s$$

$$\Diamond^* s \equiv s \vee \Diamond^+ s$$

$$\neg\Box^* s \equiv \Diamond^* \neg s$$

$$\neg\Diamond^* s \equiv \Box^* \neg s$$

Satisfiability and Demos

- A formula is satisfiable if it holds at some state of some model
- Our decision procedures search for syntactic models called demos
- A formula is satisfiable iff it is satisfied
  by a demo obtained from its subformulas
- Finite search space: Given a formula,
  only finitely many demos need to be considered

Satisfiability and Demos

- A formula is satisfiable if it holds at some state of some model
- Our decision procedures search for syntactic models called demos
- A formula is satisfiable iff it is satisfied
  by a demo obtained from its subformulas
- Finite search space: Given a formula,
  only finitely many demos need to be considered
- The states of a demo are finite sets of formulas called clauses

Satisfiability and Demos

- A formula is satisfiable if it holds at some state of some model
- Our decision procedures search for syntactic models called demos
- A formula is satisfiable iff it is satisfied
  by a demo obtained from its subformulas
- Finite search space: Given a formula,
  only finitely many demos need to be considered
- The states of a demo are finite sets of formulas called clauses
- A state of a demo satisfies every formula contained in it

Demo Search for $\Diamond^{+}\neg p \wedge \Box p \wedge \Box\Box p$

$$\Diamond^{+}\neg p,\ \Box p,\ \Box\Box p$$

Demo Search for $\Diamond^+\neg p \land \Box p \land \Box\Box p$

$$\begin{array}{l} \Diamond^+\neg p,\ \Box p,\ \Box\Box p \\ \downarrow \\ \Diamond^+\neg p,\ p,\ \Box p \end{array}$$

Demo Search for $\Diamond^{+}\neg p \wedge \Box p \wedge \Box\Box p$

$$
\begin{array}{l}
\Diamond^{+}\neg p,\ \Box p,\ \Box\Box p \\
\downarrow \\
\Diamond^{+}\neg p,\ p,\ \Box p \\
\downarrow \\
\Diamond^{+}\neg p,\ p
\end{array}
$$

Demo Search for $\Diamond^+\neg p \wedge \Box p \wedge \Box\Box p$

$$
\begin{array}{l}
\Diamond^+\neg p,\ \Box p,\ \Box\Box p \\
\downarrow \\
\Diamond^+\neg p,\ p,\ \Box p \\
\downarrow \\
\Diamond^+\neg p,\ p \\
\downarrow \\
\neg p
\end{array}
$$

Demo Search with Nominals

$$\Diamond^{+}\neg p,\ \Box(x \wedge p),\ \Diamond\Box p$$

Demo Search with Nominals

$$\Diamond^+\neg p,\ \Box(x \wedge p),\ \Diamond\Box p$$

$$\downarrow$$

$$\Diamond^+\neg p,\ x,\ p$$

Demo Search with Nominals

$$\Diamond^{+}\neg p,\ \Box(x \wedge p),\ \Diamond\Box p$$

$$\downarrow \qquad\qquad \swarrow$$

$$\Diamond^{+}\neg p,\ x,\ p,\ \Box p$$

Demo Search with Nominals

$$
\begin{tikzcd}
\Diamond^{+}\neg p, \arrow[d] & \Box(x \wedge p), & & \Diamond\Box p \arrow[dl] \\
\Diamond^{+}\neg p, \arrow[d] & x,\ \ p, & \Box p & \\
\Diamond^{+}\neg p,\ \ p & & &
\end{tikzcd}
$$

Demo Search with Nominals

$$\begin{array}{l}
\Diamond^{+}\neg p,\ \Box(x \wedge p),\ \Diamond\Box p \\
\downarrow \\
\Diamond^{+}\neg p,\ x,\ p,\ \Box p \\
\downarrow \\
\Diamond^{+}\neg p,\ p \\
\downarrow \\
\neg p
\end{array}$$

Cyclic Demo

$$G := \Box^*(\Diamond^* p \wedge \Diamond^* \neg p)$$

- Every reachable state can reach both p and $\neg p$
- Every finite model must cycle

Cyclic Demo

$$G := \Box^*(\Diamond^* p \wedge \Diamond^* \neg p)$$

- Every reachable state can reach both p and $\neg p$
- Every finite model must cycle

$$\Diamond^+ p,\ \Box G,\ \neg p$$

Cyclic Demo

$$G := \Box^*(\Diamond^* p \wedge \Diamond^* \neg p)$$

- Every reachable state can reach both p and $\neg p$
- Every finite model must cycle

$$\begin{array}{l} \Diamond^+ p,\ \Box G,\ \neg p \\ \downarrow \\ p,\ \Box G,\ \Diamond^+ \neg p \end{array}$$

Cyclic Demo

$$G := \Box^*(\Diamond^* p \land \Diamond^* \neg p)$$

- Every reachable state can reach both p and $\neg p$
- Every finite model must cycle

$$\begin{tikzcd} \Diamond^+ p,\ \Box G,\ \neg p \arrow[d, shift right] \\ p,\ \Box G,\ \Diamond^+ \neg p \arrow[u, shift right] \end{tikzcd}$$

Plan of the Talk

Foundations for efficient decision procedures
for modal logics with eventualities and nominals

1. Hintikka demos and pruning
2. Expansion and graph search
3. Backtracking search

In each part, nominals will first be ignored
and then be added in a second step

I Hintikka Demos and Pruning

- Basic theory we need
- Pruning yields complexity-optimal decision procedure

- Fischer and Ladner 1977 (PDL)
- Pratt 1979 (PDL)
- Emerson and Halpern 1985 (CTL)

- Kaminski, Schneider, and Smolka, Tableaux 2011
  (PDL with nominals, difference, and converse)

Formula Decomposition and Finite Syntactic Closure

- A formula is either conjunctive (α), disjunctive (β), or literal

$$s, t ::= \underbrace{s \wedge t \mid \Box^* s}_{\text{conjunctive}} \mid \underbrace{s \vee t \mid \Diamond^* s}_{\text{disjunctive}} \mid \underbrace{p \mid \neg p \mid \Box s \mid \Diamond s}_{\text{literal}}$$

Formula Decomposition and Finite Syntactic Closure

- A formula is either conjunctive (α), disjunctive (β), or literal

$$s, t ::= \underbrace{s \wedge t \mid \Box^* s}_{\text{conjunctive}} \mid \underbrace{s \vee t \mid \Diamond^* s}_{\text{disjunctive}} \mid \underbrace{p \mid \neg p \mid \Box s \mid \Diamond s}_{\text{literal}}$$

- Compatible formula decomposition

$s \wedge t$	α	$s,\ t$
$s \vee t$	β	$s,\ t$
$\Box^* s$	α	$s,\ \Box^+ s$
$\Diamond^* s$	β	$s,\ \Diamond^+ s$
$\Box s$	μ	s
$\Diamond s$	μ	s

Formula Decomposition and Finite Syntactic Closure

- A formula is either conjunctive (α), disjunctive (β), or literal

$$s,t ::= \underbrace{s \wedge t \mid \Box^* s}_{\text{conjunctive}} \mid \underbrace{s \vee t \mid \Diamond^* s}_{\text{disjunctive}} \mid \underbrace{p \mid \neg p \mid \Box s \mid \Diamond s}_{\text{literal}}$$

- Compatible formula decomposition

$s \wedge t$	α	$s,\ t$
$s \vee t$	β	$s,\ t$
$\Box^* s$	α	$s,\ \Box^+ s$
$\Diamond^* s$	β	$s,\ \Diamond^+ s$
$\Box s$	μ	s
$\Diamond s$	μ	s

- A denotes finite set of formulas
- $\overline{A}$ is decomposition closure of A
- Size of $\overline{A}$ is linear in the size of A
- α-β-decomposition terminates
- μ is modal literal
- $\Box^+ s := \Box\Box^* s$
- $\Diamond^+ s := \Diamond\Diamond^* s$

Hintikka Sets

- A is a Hintikka set if
  1. If $\neg p \in A$, then $p \notin A$
  2. If $s \in A$ is of type α, then all constituents of s are in A
  3. If $s \in A$ is of type β, then at least one constituent of s is in A
- $H \equiv \mathcal{L}H$ if H Hintikka set ($\mathcal{L}H$ is set of literals in H)
- Every state w of every model $\mathcal{M}$ yields a finite Hintikka set in $\overline{A}$:

$$H_w := \{\, s \in \overline{A} \mid \mathcal{M}, w \models s \,\}$$

- The Hintikka sets for $\mathcal{M}$ and A contain exactly the formulas $s \in \overline{A}$ that are satisfiable in $\mathcal{M}$

Hintikka Systems

- A Hintikka system is a set of Hintikka sets
- A Hintikka system $\mathcal{H}$ describes a model:
  - The states are the sets $H \in \mathcal{H}$
  - A state H is labeled with p iff $p \in H$
  - There is an edge $H \to H'$ iff $\{\, s \mid \Box s \in H \,\} \subseteq H'$
- We call $\mathcal{R}H := \{\, s \mid \Box s \in H \,\}$ the request of H

Demos

- A demo is a Hintikka system $\mathcal{H}$ such that:
  - If $\Diamond s \in H \in \mathcal{H}$, then $\exists\, H' \in \mathcal{H}.\ H \rightarrow_{\mathcal{H}} H'$ and $s \in H'$
  - If $\Diamond^{+} s \in H \in \mathcal{H}$, then $\exists\, H' \in \mathcal{H}.\ H \rightarrow_{\mathcal{H}}^{+} H'$ and $s \in H'$
- Lemma

  $\mathcal{H}, H \models s$ if $\mathcal{H}$ is a demo and $s \in H \in \mathcal{H}$
- Small Model Theorem

  $s \in \overline{A}$ satisfiable $\iff$ $\exists$ demo $\mathcal{H} \in 2^{2^{\overline{A}}}$ $\exists H \in \mathcal{H}.\ s \in H$
- Naive decision procedure is double exponential

Pruning (Pratt 1979)

- Start from maximal Hintikka system over A
- Stepwise delete Hintikka sets that violate a demo condition
- Terminates with largest demo over A
    - demos are closed under union

Pruning (Pratt 1979)

- Start from maximal Hintikka system over A
- Stepwise delete Hintikka sets that violate a demo condition
- Terminates with largest demo over A
  - demos are closed under union
- Yields exponential decision procedure
  - complexity-optimal
  - decides satisfiability of every formula $s \in \overline{A}$ in one go
  - not practical since it starts with all Hintikka sets

Pruning Generalizes

- Pruning can be applied to every Hintikka system
- Always terminates with largest demo

Pruning Generalizes

- Pruning can be applied to every Hintikka system
- Always terminates with largest demo
- Correctness: $\mathcal{H}_0 \to \mathcal{H}_1 \to \mathcal{H}_2 \to \cdots \to \mathcal{H}_n$
  1. $\mathcal{H}_0 \supset \mathcal{H}_1 \supset \mathcal{H}_2 \supset \cdots \supset \mathcal{H}_n$
  2. $\mathcal{H}_n$ is a demo
  3. $\mathcal{H}_{k+1}$ contains all demos contained in $\mathcal{H}_k$
  4. $\mathcal{H}_n$ is greatest demo contained in $\mathcal{H}_0$

Everything Extends to Nominals

- Require Hintikka systems to be nominally coherent
  (every nominal appears in at most one Hintikka set)
- Exponentially many maximal Hintikka systems
- Pruning still yields exponential decision procedure
  (every demo is contained in a maximal Hintikka system)

II Expansion and Graph Search

- Represent Hintikka sets by their literals
- Start from input clauses
- Stepwise add clauses by expansion
- Determine satisfiable and unsatisfiable clauses
- Stop if input clauses are determined

- Kaminski and Smolka 2010, 2011

- Related to Goré and Widmann's [2009, 2010] decision procedure for PDL with converse

Clauses = Literal Hintikka Sets

- A literal is a formula that is neither α nor β
  (p, $\neg p$, $\Box s$, $\Diamond s$)

Clauses = Literal Hintikka Sets

- A literal is a formula that is neither α nor β (p, $\neg p$, $\Box s$, $\Diamond s$)
- The non-literal formulas of a Hintikka set are logically redundant ($H \equiv \mathcal{L}H$)

Clauses = Literal Hintikka Sets

- A literal is a formula that is neither α nor β
  (p, $\neg p$, $\Box s$, $\Diamond s$)
- The non-literal formulas of a Hintikka set are logically redundant
  ($H \equiv \mathcal{L}H$)
- A clause is a Hintikka set just containing literals
  (i.e., a conflict-free set of literals)

Clauses = Literal Hintikka Sets

- A literal is a formula that is neither α nor β (p, $\neg p$, $\Box s$, $\Diamond s$)
- The non-literal formulas of a Hintikka set are logically redundant ($H \equiv \mathcal{L}H$)
- A clause is a Hintikka set just containing literals (i.e., a conflict-free set of literals)
- Support: $C \triangleright A :\iff \exists$ Hintikka set $H.\ A \subseteq H \land \mathcal{L}H \subseteq C$

Clausal Demos

- A clause system is a set of clauses
- Every clause system $\mathcal{S}$ comes with a transition relation $\to_{\mathcal{S}}$:

$$C \to_{\mathcal{S}} C' :\iff C \in \mathcal{S} \wedge C' \in \mathcal{S} \wedge C' \triangleright \mathcal{R}C$$

- Clausal demo: set $\mathcal{S}$ of clauses satisfying the demo conditions:
  - If $\Diamond s \in C \in \mathcal{S}$, then $\exists\, C' \in \mathcal{S}.\ \ C \to_{\mathcal{S}} C'$ and $C' \triangleright s$
  - If $\Diamond^+ s \in C \in \mathcal{S}$, then $\exists\, C' \in \mathcal{S}.\ \ C \to_{\mathcal{S}}^+ C'$ and $C' \triangleright s$
- Hintikka demo yields clausal demo and vice versa

Demonstrated Clauses

- Let $\mathcal{S}$ be a clause system
- A clause C is demonstrated in $\mathcal{S}$ if $\exists$ demo $\mathcal{D} \subseteq \mathcal{S}.\ C \in \mathcal{D}$
- Demonstrated clauses are satisfiable
- The clauses demonstrated in $\mathcal{S}$ can be computed by pruning

Disjunctive Normal Forms (DNF)

- Sets of formulas are interpreted conjunctively
  i.e., $\{s_1, \dots, s_n\}$ is interpreted as $s_1 \wedge \dots \wedge s_n$
- A set $\{C_1, \dots, C_n\}$ of clauses is a DNF of A if
  - $A \equiv C_1 \vee \dots \vee C_n$
  - $C_i \rhd A$ and $C_i \subseteq \overline{A}$ for all $i \in \{1, \dots, n\}$
- Every finite A has a DNF, can be obtained by α-β-decomposition
- DNFs are not unique
- If A has an empty DNF, then A is unsatisfiable

Example of DNF Computation

$$\Diamond^*(p \wedge q),\ \Box^*(\neg p \vee \neg q)$$

Example of DNF Computation

$$\Diamond^*(p \wedge q),\ \Box^*(\neg p \vee \neg q)$$

$$\neg p \vee \neg q$$

$$\color{red}{\Box^+(\neg p \vee \neg q)}$$

$$\begin{array}{c|c}
\begin{array}{c} p \wedge q \\ p \\ q \\ \hline \begin{array}{c|c} \neg p & \neg q \end{array} \end{array} &
\begin{array}{c} \color{red}{\Diamond^+(p \wedge q)} \\ \hline \begin{array}{c|c} \color{red}{\neg p} & \color{red}{\neg q} \end{array} \end{array}
\end{array}$$

Example of DNF Computation

$$\Diamond^*(p \wedge q),\ \Box^*(\neg p \vee \neg q)$$

$$\neg p \vee \neg q$$

$$\color{red}{\Box^+(\neg p \vee \neg q)}$$

$$\begin{array}{c|c}
\begin{array}{c} p \wedge q \\ p \\ q \\ \hline \begin{array}{c|c} \neg p & \neg q \\ \otimes & \otimes \end{array} \end{array} & \begin{array}{c} \color{red}{\Diamond^+(p \wedge q)} \\ \hline \begin{array}{c|c} \color{red}{\neg p} & \color{red}{\neg q} \end{array} \end{array}
\end{array}$$

Example of DNF Computation

$$\Diamond^*(p \wedge q),\ \Box^*(\neg p \vee \neg q)$$

$$\neg p \vee \neg q$$

$$\Box^+(\neg p \vee \neg q)$$

$$
\begin{array}{c|c}
\begin{array}{c}
p \wedge q \\
p \\
q \\
\hline
\begin{array}{c|c} \neg p & \neg q \end{array} \\
\otimes \quad \otimes
\end{array}
&
\begin{array}{c}
\Diamond^+(p \wedge q) \\
\hline
\begin{array}{c|c} \neg p & \neg q \end{array} \\
C_1 \quad C_2
\end{array}
\end{array}
$$

Expansions

- $\Diamond s \in C$ needs a successor clause in a demo
- An expansion of $\Diamond s \in C$ is a set of possible successor clauses
- If a set $\mathcal{S}$ contains an expansion for every $\Diamond s \in C$,
  then it contains a demo for every satisfiable clause
- expansion of $\Diamond s \in C$ if s is not an eventuality
  DNF of $\{s\} \cup \mathcal{R}C$
- expansion of $\Diamond^+ s \in C$
  (DNF of $\{s\} \cup \mathcal{R}C$) $\cup$ (DNF of $\{\Diamond^+ s\} \cup \mathcal{R}C$)

Expansions

- Expansion of $\{\Diamond p,\ \Box(q_1 \vee q_2)\}$:
  $\{p, q_1\}$
  $\{p, q_2\}$

Expansions

- Expansion of $\{\Diamond p,\ \Box(q_1 \vee q_2)\}$:
  $\{p, q_1\}$
  $\{p, q_2\}$
- Expansion of $\{\Diamond^+ p,\ \Box^+ \Diamond^+ p\}$:
  $\{p, \Diamond^+ p, \Box^+ \Diamond^+ p\}$
  $\{\Diamond^+ p, \Box^+ \Diamond^+ p\}$

Expansions

- Expansion of $\{\Diamond p,\ \Box(q_1 \vee q_2)\}$:
  $\{p, q_1\}$
  $\{p, q_2\}$
- Expansion of $\{\Diamond^+ p,\ \Box^+\Diamond^+ p\}$:
  $\{p, \Diamond^+ p, \Box^+\Diamond^+ p\}$
  $\{\Diamond^+ p, \Box^+\Diamond^+ p\}$
- If a state w satisfies C and w' is a successor of w satisfying s, then w' satisfies some clause in every expansion of $\Diamond^+ s \in C$

$$\begin{array}{ccc} \Diamond^+ s \in C & & s \\ & & \\ w & \to_{\mathcal{M}} & w' \end{array}$$

Expansions

- Expansion of $\{\Diamond p,\ \Box(q_1 \vee q_2)\}$:
  $\{p, q_1\}$
  $\{p, q_2\}$
- Expansion of $\{\Diamond^+ p,\ \Box^+\Diamond^+ p\}$:
  $\{p, \Diamond^+ p, \Box^+\Diamond^+ p\}$
  $\{\Diamond^+ p, \Box^+\Diamond^+ p\}$
- If a state w satisfies C and w' is a successor of w satisfying s, then w' satisfies some clause in every expansion of $\Diamond^+ s \in C$

$$\Diamond^+ s \in C \quad \rightarrow_{\mathcal{S}} \quad C' \quad \triangleright \quad s$$

$$w \quad \rightarrow_{\mathcal{M}} \quad w'$$

Saturation under Expansion

- A set of clauses $\mathcal{S}$ is saturated if it contains an expansion for every $\Diamond s \in C \in \mathcal{S}$
- Theorem Let $\mathcal{S}$ be saturated. Then: $C \in \mathcal{S}$ is satisfiable iff C is demonstrated in $\mathcal{S}$

Saturation under Expansion

- A set of clauses $\mathcal{S}$ is saturated if it contains an expansion for every $\Diamond s \in C \in \mathcal{S}$
- Theorem Let $\mathcal{S}$ be saturated. Then: $C \in \mathcal{S}$ is satisfiable iff C is demonstrated in $\mathcal{S}$
- Decision procedure
  1. Start with $\mathcal{S} := \{C_0\}$
  2. Expand $\mathcal{S}$ until either C_0 demonstrated in $\mathcal{S}$ or $\mathcal{S}$ saturated

Saturation under Expansion

- A set of clauses $\mathcal{S}$ is saturated if it contains an expansion for every $\Diamond s \in C \in \mathcal{S}$
- Theorem Let $\mathcal{S}$ be saturated. Then: $C \in \mathcal{S}$ is satisfiable iff C is demonstrated in $\mathcal{S}$
- Decision procedure
  1. Start with $\mathcal{S} := \{C_0\}$
  2. Expand $\mathcal{S}$ until either C_0 demonstrated in $\mathcal{S}$ or $\mathcal{S}$ saturated
- Can be efficient if C is satisfiable
- Inefficient if C is unsatisfiable
- How can we determine unsatisfiable clauses?

Sinks

- A sink is a sufficiently expanded set of clauses so that it is clear that some eventuality cannot be fulfilled

Sinks

- A sink is a sufficiently expanded set of clauses so that it is clear that some eventuality cannot be fulfilled
- A sink is a pair $(\mathcal{S}, \Diamond^+ s)$ such that every clause $C \in \mathcal{S}$ satisfies
  1. $\Diamond^+ s \in C$ and $C \not\triangleright s$
  2. $\mathcal{S}$ contains every satisfiable clause of an expansion of $\Diamond^+ s \in C$

Sinks

- A sink is a sufficiently expanded set of clauses so that it is clear that some eventuality cannot be fulfilled
- A sink is a pair $(\mathcal{S}, \Diamond^+ s)$ such that every clause $C \in \mathcal{S}$ satisfies
  1. $\Diamond^+ s \in C$ and $C \not\triangleright s$
  2. $\mathcal{S}$ contains every satisfiable clause of an expansion of $\Diamond^+ s \in C$
- Theorem   Every clause of a sink is unsatisfiable

Sinks

- A sink is a sufficiently expanded set of clauses so that it is clear that some eventuality cannot be fulfilled
- A sink is a pair $(\mathcal{S}, \Diamond^+ s)$ such that every clause $C \in \mathcal{S}$ satisfies
  1. $\Diamond^+ s \in C$ and $C \not\triangleright s$
  2. $\mathcal{S}$ contains every satisfiable clause of an expansion of $\Diamond^+ s \in C$
- Theorem   Every clause of a sink is unsatisfiable
- Example of a 2-clause sink

$G := (\neg p \vee \neg q)$

$$\begin{tikzcd}
\Diamond^+(p \wedge q),\ \Box^+ G,\ \neg p \arrow[loop left] \arrow[d, leftrightarrow] \\
\Diamond^+(p \wedge q),\ \Box^+ G,\ \neg q \arrow[loop left]
\end{tikzcd}$$

Sinks

- A sink is a sufficiently expanded set of clauses so that it is clear that some eventuality cannot be fulfilled
- A sink is a pair $(\mathcal{S}, \Diamond^+ s)$ such that every clause $C \in \mathcal{S}$ satisfies
  1. $\Diamond^+ s \in C$ and $C \not\triangleright s$
  2. $\mathcal{S}$ contains every satisfiable clause of an expansion of $\Diamond^+ s \in C$
- Theorem   Every clause of a sink is unsatisfiable
- Example of a 2-clause sink

$$G := (\neg p \vee \neg q) \wedge \Diamond D$$

$$\begin{tikzcd} \Diamond^+(p \wedge q),\ \Box^+ G,\ \neg p,\ \Diamond D \arrow[loop above] \arrow[d, leftrightarrow] \\ \Diamond^+(p \wedge q),\ \Box^+ G,\ \neg q,\ \Diamond D \arrow[loop below] \end{tikzcd}$$

Refutation

1. If every clause of an expansion of $\Diamond s \in C$ is unsatisfiable, then C is unsatisfiable
2. Every superset of an unsatisfiable clause is unsatisfiable
3. Every clause of a sink is unsatisfiable

Theorem

If $\mathcal{S}$ is saturated, then the refutation rules can refute every unsatisfiable clause in $\mathcal{S}$.

Incremental Decision Procedure

1. Maintains a set $\mathcal{S}$ of a clauses and two disjoint subsets $\mathcal{D} \subseteq \mathcal{S}$ and $\mathcal{R} \subseteq \mathcal{S}$ of demonstrated and refuted clauses

$$\underbrace{\boxed{\mathcal{D}}\boxed{\quad\quad}\boxed{\mathcal{R}}}_{\mathcal{S}}$$

2. A clause is determined if it is in $\mathcal{D}$ or $\mathcal{R}$
3. Starts with $\mathcal{D} := \mathcal{R} := \emptyset$
4. Expands undetermined clauses in $\mathcal{S}$
5. Grows $\mathcal{D}$ and $\mathcal{R}$ by pruning and refutation
6. Stops once initial clauses are determined

- Complexity-optimal
- Can be efficient for clauses with small refutations or small demos

Extension to Nominals

- Additional expansion rule that adds unions of nominal clauses
- Soundness of demonstration and refutation is preserved
- Completeness of demonstration is preserved
  (prune every maximal nominally coherent subsystem)
- Completeness of refutation is lost

III Backtracking Search

- Simple algorithm
- Works well in presence of nominals
- DL reasoners for expressive logics are backtracking

- Kaminski and Smolka 2010 (IJCAR)

Functional Demos

A functional demo is a clause system that

$$\begin{array}{l}
\Diamond^{+}\neg p,\ \Box p,\ \Box\Box p \\
\downarrow \\
\Diamond^{+}\neg p,\ p,\ \Box p \\
\downarrow \\
\Diamond^{+}\neg p,\ p \\
\downarrow \\
\neg p
\end{array}$$

- fixes a unique successor for every diamond by a link
- contains no delegation loop (red links are delegating) (ensures satisfaction of eventualities)

Backtracking Demo Search

- Construct a functional demo by single successor expansion of diamonds (don't know choice)
- Every clause added is part of the final demo
  - Fail if a diamond has no successor
  - Fail if a delegation loop is introduced
- Succeeds with functional demo iff input clause is satisfiable
- Minimal bureaucracy (no pruning, no refutation)
- Extends smoothly to nominals
- Worst-case runtime is double exponential

Example

$$\Diamond^{+}\neg p,\ \Box p,\ \Box\Box p$$

Example

$$\Diamond^{+}\neg p,\ \Box p,\ \Box\Box p$$
$$\downarrow$$
$$\Diamond^{+}\neg p,\ p,\ \Box p$$

Example

$$\Diamond^{+}\neg p,\ \Box p,\ \Box\Box p$$

$$\downarrow$$

$$\Diamond^{+}\neg p,\ p,\ \Box p$$

$$\downarrow$$

$$\Diamond^{+}\neg p,\ p$$

Example

$$\Diamond^{+}\neg p,\ \Box p,\ \Box\Box p$$

$$\downarrow$$

$$\Diamond^{+}\neg p,\ p,\ \Box p$$

$$\downarrow$$

$$\Diamond^{+}\neg p,\ p$$

$$\downarrow$$

$$\neg p$$

Success

Example

$$\Diamond^{+}\neg p,\ \Box p,\ \Box\Box p$$

$$\downarrow$$

$$\Diamond^{+}\neg p,\ p,\ \Box p$$

$$\downarrow$$

$$\Diamond^{+}\neg p,\ p$$

$$\searrow$$

$$\Diamond^{+}\neg p$$

Example

$$\Diamond^{+}\neg p,\ \Box p,\ \Box\Box p$$

$$\downarrow$$

$$\Diamond^{+}\neg p,\ p,\ \Box p$$

$$\downarrow$$

$$\Diamond^{+}\neg p,\ p$$

$$\searrow$$

$$\Diamond^{+}\neg p$$

$$\swarrow$$

$$\neg p$$

Success

Example

$$\Diamond^+\neg p,\ \Box p,\ \Box\Box p$$

$$\Diamond^+\neg p,\ p,\ \Box p$$

$$\Diamond^+\neg p,\ p$$

$$\Diamond^+\neg p$$

Failure because of delegation loop

Completeness Argument

- Why is a demo found if the initial clause is satisfiable?
- Clauses added by expansion are satisfied by initial model
- Every proper delegation link $C(\Diamond^+ s)D$ reduces δ-distance:
  1. $\delta_{\mathcal{M}} Ds > 0$
  2. $\delta_{\mathcal{M}} Cs = 0$ or $\delta_{\mathcal{M}} Cs > \delta_{\mathcal{M}} Ds$
- $\delta_{\mathcal{M}} As :=$ minimal distance
  from a state satisfying A to a state satisfying s
- Minimal distance idea appears in [Baader 1990]

Extension to Nominals

- Start with nominally coherent set of clauses
- If an expansion step introduces a nominal clause, regain nominal coherence as follows:
  - Union new clause with all clauses containing a common nominal
  - Fail if union is not a clause; otherwise:
  - Redirect links pointing to subsumed clauses to union clause
  - Delete subsumed clauses

Example

$$\Diamond^{+} p,\ \neg p,\ \Box(x \land \neg p),\ \Diamond\Box\neg p$$

Example

$$\Diamond^+ p,\ \neg p,\ \Box(x \wedge \neg p),\ \Diamond\Box\neg p$$

$$\downarrow$$

$$x,\ \Diamond^+ p,\ \neg p$$

Example

$$\Diamond^+ p,\ \neg p,\ \Box(x \wedge \neg p),\ \Diamond\Box\neg p$$

$$\downarrow$$

$$x,\ \Diamond^+ p,\ \neg p$$

$$\downarrow$$

$$p$$

Example

$$\Diamond^+ p,\ \neg p,\ \Box(x \wedge \neg p),\ \Diamond\Box\neg p$$

$$x,\ \Diamond^+ p,\ \neg p \qquad\qquad x,\ \neg p,\ \Box\neg p$$

$$p$$

Nominal union needed

Example

$$\Diamond^+ p,\ \neg p,\ \Box(x \wedge \neg p),\ \Diamond\Box\neg p$$

$$x,\ \Diamond^+ p,\ \neg p \qquad \Diamond^+ p,\ x,\ \neg p,\ \Box\neg p$$

$$p$$

Delete subsumed nominal clause

Example

$$\Diamond^{+}p,\ \neg p,\ \Box(x \wedge \neg p),\ \Diamond\Box\neg p$$

$$\Diamond^{+}p,\ x,\ \neg p,\ \Box\neg p$$

p

Example

$$
\begin{tikzcd}
\Diamond^+ p,\ \neg p,\ \Box(x \wedge \neg p),\ \Diamond\Box\neg p \arrow[dr, red] \arrow[dr, black] & \\
& \Diamond^+ p,\ {\color{blue}x},\ \neg p,\ \Box\neg p \arrow[d, red] \\
p & \Diamond^+ p,\ \neg p
\end{tikzcd}
$$

Example

$$
\begin{tikzcd}
& \Diamond^+ p,\ \neg p,\ \Box(x \wedge \neg p),\ \Diamond\Box\neg p \arrow[d] \\
& \Diamond^+ p,\ x,\ \neg p,\ \Box\neg p \arrow[d] \\
p & \Diamond^+ p,\ \neg p \arrow[l]
\end{tikzcd}
$$

Conclusions

- Decision procedures for $\mathcal{ALC}$ with eventualities and nominals
- Approach: Goal-directed demo search
- Graph search and backtracking search
  - Backtracking search easier to implement
  - Graph search has not been implemented with nominals
  - Impressive graph prover for PDL by Goré and Widmann
  - Backtracking search performs best on most K benchmarks
- Everything extends to PDL (union and composition of roles)
  - Fischer-Ladner decomposition does not satisfy $H \equiv \mathcal{L}H$ (e.g., $\{\langle a^{**}\rangle\bot,\ \langle a^{*}\rangle\langle a^{**}\rangle\bot\}$ is an unsatisfiable Hintikka set)
  - Expansion requires terminating α-β-decomposition (can be arranged)
- Open: Extension to converse roles