

Model checking of deontic interpreted systems via BDD's

Alessio Lomuscio
alessio@dcs.kcl.ac.uk

Joint work with Franco Raimondi, Marek Sergot.

Liverpool - July04

Overview

Overview

- Reasoning about faults and/or incorrect behaviour in MAS.

Overview

- Reasoning about faults and/or incorrect behaviour in MAS.
- Deontic interpreted systems (Lomuscio and Sergot).

Overview

- Reasoning about faults and/or incorrect behaviour in MAS.
- Deontic interpreted systems (Lomuscio and Sergot).
- Symbolic Model checking via bdds for deontic interpreted systems (Raimondi and Lomuscio).

Overview

- Reasoning about faults and/or incorrect behaviour in MAS.
- Deontic interpreted systems (Lomuscio and Sergot).
- Symbolic Model checking via bdds for deontic interpreted systems (Raimondi and Lomuscio).
- Dining cryptographers with faults.

Motivation

Motivation

They may be programmed by several different software houses. Not all of them may wish to have their agent to follow all the prescribed specification for the system.

For verification purposes we would wish to check:

Motivation

They may be programmed by several different software houses. Not all of them may wish to have their agent to follow all the prescribed specification for the system.

For verification purposes we would wish to check:

- Properties of the system that hold *irrespective of the behaviour of the agents*.

Motivation

They may be programmed by several different software houses. Not all of them may wish to have their agent to follow all the prescribed specification for the system.

For verification purposes we would wish to check:

- Properties of the system that hold *irrespective of the behaviour of the agents*.
- Properties that hold *only if* all the agents of the system behave as intended.

Motivation

They may be programmed by several different software houses. Not all of them may wish to have their agent to follow all the prescribed specification for the system.

For verification purposes we would wish to check:

- Properties of the system that hold *irrespective of the behaviour of the agents*.
- Properties that hold *only if* all the agents of the system behave as intended.
- Properties that hold *only if* some the agents of the system behave as intended.

Deontic interpreted systems

Deontic interpreted systems

Deontic interpreted systems are a semantics to reason about time, knowledge, and correct functioning behaviour.

Deontic interpreted systems

Deontic interpreted systems are a semantics to reason about time, knowledge, and correct functioning behaviour.

- They extend standard interpreted systems (Halpern et al):

$$S \subseteq L_e \times L_1 \times \cdots \times L_n.$$

Deontic interpreted systems

Deontic interpreted systems are a semantics to reason about time, knowledge, and correct functioning behaviour.

- They extend standard interpreted systems (Halpern et al):
 $S \subseteq L_e \times L_1 \times \cdots \times L_n$.
- Assume some local states are “allowed” or green, others are “disallowed” or red. Intuitively these will represent “correct/incorrect” functioning behaviour of the agents.

Deontic interpreted systems

Deontic interpreted systems are a semantics to reason about time, knowledge, and correct functioning behaviour.

- They extend standard interpreted systems (Halpern et al):
 $S \subseteq L_e \times L_1 \times \cdots \times L_n$.
- Assume some local states are “allowed” or green, others are “disallowed” or red. Intuitively these will represent “correct/incorrect” functioning behaviour of the agents.
 - Green states: $L_e \supseteq G_e, \dots, L_n \supseteq G_n, G_i \neq \emptyset$.

Deontic interpreted systems

Deontic interpreted systems are a semantics to reason about time, knowledge, and correct functioning behaviour.

- They extend standard interpreted systems (Halpern et al):
 $S \subseteq L_e \times L_1 \times \cdots \times L_n$.
- Assume some local states are “allowed” or green, others are “disallowed” or red. Intuitively these will represent “correct/incorrect” functioning behaviour of the agents.
 - Green states: $L_e \supseteq G_e, \dots, L_n \supseteq G_n, G_i \neq \emptyset$.
 - Red states: $R_i = L_i \setminus G_i$.

Deontic interpreted systems

Deontic interpreted systems are a semantics to reason about time, knowledge, and correct functioning behaviour.

- They extend standard interpreted systems (Halpern et al):
 $S \subseteq L_e \times L_1 \times \cdots \times L_n$.
- Assume some local states are “allowed” or green, others are “disallowed” or red. Intuitively these will represent “correct/incorrect” functioning behaviour of the agents.
 - Green states: $L_e \supseteq G_e, \dots, L_n \supseteq G_n, G_i \neq \emptyset$.
 - Red states: $R_i = L_i \setminus G_i$.
 - An agent is in a red local state if it is in violation of its specification (crashes, memory violation, or rationality if the agents are players in a game theoretical setting).

Deontic interpreted systems

Deontic interpreted systems are a semantics to reason about time, knowledge, and correct functioning behaviour.

- They extend standard interpreted systems (Halpern et al):
 $S \subseteq L_e \times L_1 \times \dots \times L_n$.
- Assume some local states are “allowed” or green, others are “disallowed” or red. Intuitively these will represent “correct/incorrect” functioning behaviour of the agents.
 - Green states: $L_e \supseteq G_e, \dots, L_n \supseteq G_n, G_i \neq \emptyset$.
 - Red states: $R_i = L_i \setminus G_i$.
 - An agent is in a red local state if it is in violation of its specification (crashes, memory violation, or rationality if the agents are players in a game theoretical setting).
- Syntax: $\phi ::= p, \neg\phi, \phi \wedge \phi, K_i\phi, O_i\phi, \widehat{K}_j^i\phi, EG\phi, E(\phi U \phi)$

Deontic systems of global states

Deontic systems of global states

- A set of local states L_i , for each agent $i = \{1, \dots, n\}$

Deontic systems of global states

- A set of local states L_i , for each agent $i = \{1, \dots, n\}$
- A non-empty set $G_i \subseteq L_i$ of “green” states.

Deontic systems of global states

- A set of local states L_i , for each agent $i = \{1, \dots, n\}$
- A non-empty set $G_i \subseteq L_i$ of “green” states.
- A set of actions Act_i .

Deontic systems of global states

- A set of local states L_i , for each agent $i = \{1, \dots, n\}$
- A non-empty set $G_i \subseteq L_i$ of “green” states.
- A set of actions Act_i .
- A protocol $P_i : L_i \rightarrow 2^{Act_i}$.

Deontic systems of global states

- A set of local states L_i , for each agent $i = \{1, \dots, n\}$
- A non-empty set $G_i \subseteq L_i$ of “green” states.
- A set of actions Act_i .
- A protocol $P_i : L_i \rightarrow 2^{Act_i}$.
- Local evolution function: $\pi_i : S \times \mathbf{Act} \rightarrow S_i$.

Deontic systems of global states

- A set of local states L_i , for each agent $i = \{1, \dots, n\}$
- A non-empty set $G_i \subseteq L_i$ of “green” states.
- A set of actions Act_i .
- A protocol $P_i : L_i \rightarrow 2^{Act_i}$.
- Local evolution function: $\pi_i : S \times \mathbf{Act} \rightarrow S_i$.
- A set of global states $S \subset L_1 \times \dots \times L_n \times L_E$.

Deontic systems of global states

- A set of local states L_i , for each agent $i = \{1, \dots, n\}$
- A non-empty set $G_i \subseteq L_i$ of “green” states.
- A set of actions Act_i .
- A protocol $P_i : L_i \rightarrow 2^{Act_i}$.
- Local evolution function: $\pi_i : S \times \mathbf{Act} \rightarrow S_i$.
- A set of global states $S \subset L_1 \times \dots \times L_n \times L_E$.
- Reachable states may be computed from a set of initial states $I \subset S$.

DIS - interpretation

Given a set of propositional variables P , a *deontic interpreted system* is a pair $DIS = (S, h)$ where $h : S \rightarrow 2^P$ is an interpretation function.

$$g \models O_i \phi \quad \text{iff} \quad \forall g' \in S, l_i(g') \in G_i \text{ implies } g' \models \phi,$$

$$g \models K_i \phi \quad \text{iff} \quad \forall g' \in S, l_i(g) = l_i(g') \text{ implies } g' \models \phi,$$

$$g \models \widehat{K}_j^i \phi \quad \text{iff} \quad \forall g' \in S, l_i(g) = l_i(g') \text{ and } l_j(g') \in G_j \\ \text{implies } g' \models \phi,$$

DIS - notes

DIS - notes

- The logic $KD45^{i-j}$ is weakly-complete wrt atemporal DIS.

DIS - notes

- The logic $KD45^{i-j}$ is weakly-complete wrt atemporal DIS.
- Various implications among $K_i O_j$, $O_i K_j$, \widehat{K}_j^i , \widehat{K}_i^j .

DIS - notes

- The logic $KD45^{i-j}$ is weakly-complete wrt atemporal DIS.
- Various implications among $K_i O_j$, $O_i K_j$, \widehat{K}_j^i , \widehat{K}_i^j .
- Tested on bit transmission problem with faults...

BDD's 4 DIS

BDD's 4 DIS

Idea: extend OBDD's-based algorithm for CTL to deontic interpreted systems.

BDD's 4 DIS

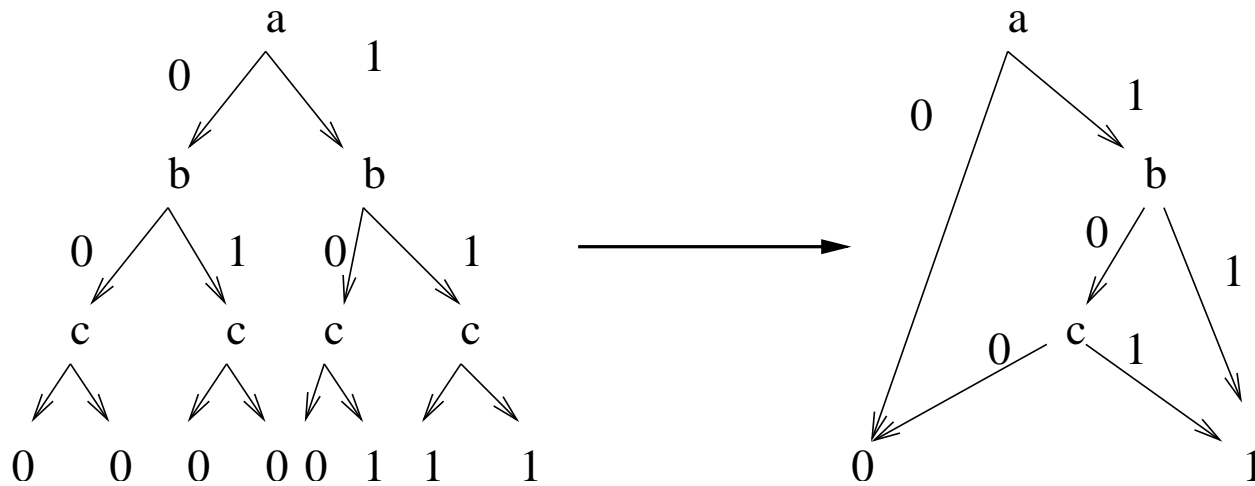
Idea: extend OBDD's-based algorithm for CTL to deontic interpreted systems.

To do this we encode all the parameters needed by the verification algorithm as boolean formulas and operate on them via OBDD's.

BDD's 4 DIS

Idea: extend OBDD's-based algorithm for CTL to deontic interpreted systems.

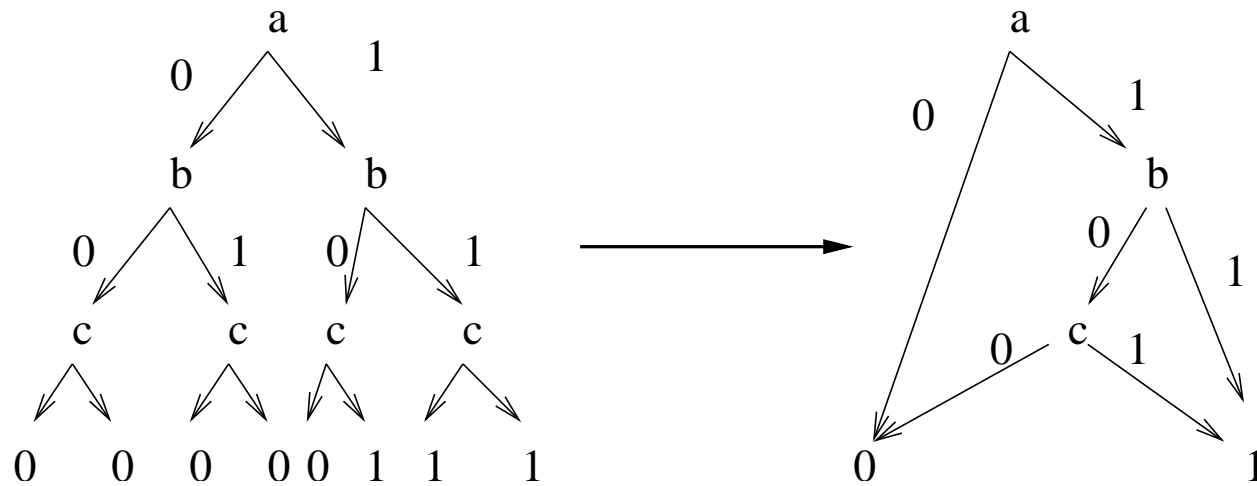
To do this we encode all the parameters needed by the verification algorithm as boolean formulas and operate on them via OBDD's.



BDD's 4 DIS

Idea: extend OBDD's-based algorithm for CTL to deontic interpreted systems.

To do this we encode all the parameters needed by the verification algorithm as boolean formulas and operate on them via OBDD's.



OBDD's have been proven to be a very efficient technique to manipulate boolean formulas.

Translating into boolean formulas

1. Encode local states by means of boolean variables.

$nv(i) = \lceil \log_2 |L_i| \rceil$ variables are needed for each agent.

$N = \sum_i nv(i)$ variables are needed to encode a global state:

$$g = (v_1, \dots, v_N).$$

2. Similarly for actions: $a = (a_1, \dots, a_M)$.

3. Protocols and evolution functions can be expressed as boolean functions involving the boolean variables above.

4. For the algorithm below, we need a boolean function for the temporal relation between two global states: $R_t(g, g')$ iff $\exists a \in Act : t(g, a, g')$ is true and each local action $a_i \in a$ is enabled by the protocol of agent i in the local state $l_i(g)$.

The algorithm

The following algorithm *SAT* computes the set of global states in which a formula ϕ holds, denoted with $[[\phi]]$

$$\begin{aligned}
& SAT(\phi) \{ \\
& \quad \phi \text{ is an atomic formula: return } h(\phi); \\
& \quad \phi \text{ is } \neg\phi_1: \text{ return } S \setminus SAT(\phi_1); \\
& \quad \phi \text{ is } \phi_1 \wedge \phi_2: \text{ return } SAT(\phi_1) \cap \\
& \quad \quad SAT(\phi_2); \\
& \quad \phi \text{ is } EX\phi_1: \text{ return } SAT_{EX}(\phi_1); \\
& \quad \phi \text{ is } E(\phi_1 U \phi_2): \text{ return } SAT_{EU}(\phi_1, \phi_2); \\
& \quad \phi \text{ is } EG\phi_1: \text{ return } SAT_{EG}(\phi_1); \\
& \quad \phi \text{ is } K_i\phi_1: \text{ return } SAT_K(\phi_1, i); \\
& \quad \phi \text{ is } O_i\phi_1: \text{ return } SAT_O(\phi_1, i); \\
& \quad \phi \text{ is } \widehat{K}_j^i \phi_1: \text{ return } SAT_{\widehat{K}_j^i}(\phi_1, i, j); \\
& \quad \}
\end{aligned}$$

The algorithm – 2

```
 $SAT_K(\phi, i) \{$   
   $X = SAT(\neg\phi);$   
   $Y = \{g \in S \mid \exists g' \in X \text{ and } R_i(g, g')\}$   
  return  $\neg Y;$   
}
```

```
 $SAT_O(\phi, i) \{$   
   $X = SAT(\neg\phi);$   
   $Y = \{g \in S \mid \exists g' \in X \text{ and } R_i^O(g, g')\}$   
  return  $\neg Y;$   
}
```

```
SATE( $\phi, \Gamma$ ) {  
  X = SAT( $\neg\phi$ );  
  Y = { $g \in G \mid R_{\Gamma}^E(g, g')$  and  $g' \in X$ }  
  return  $\neg Y$ ;  
}
```

```
SATC( $\phi, \Gamma$ ) {  
  X = SAT( $\phi$ );  
  Y = G;  
  while ( X  $\neq$  Y ) {  
    X = Y;  
    Y = { $g \in G \mid R_{\Gamma}^E(g, g')$  and  $g' \in Y$  and  $g' \in \text{SAT}(\phi)$ }  
  }  
  return Y;  
}
```

$SAT_{KH}(\phi, i, j) \{$

$X = SAT(\neg\phi);$

$Y = \{g \in S | \exists g' \in X \text{ and } R_i(g, g') \text{ and } R_j^O(g, g')\}$

return $\neg Y$;

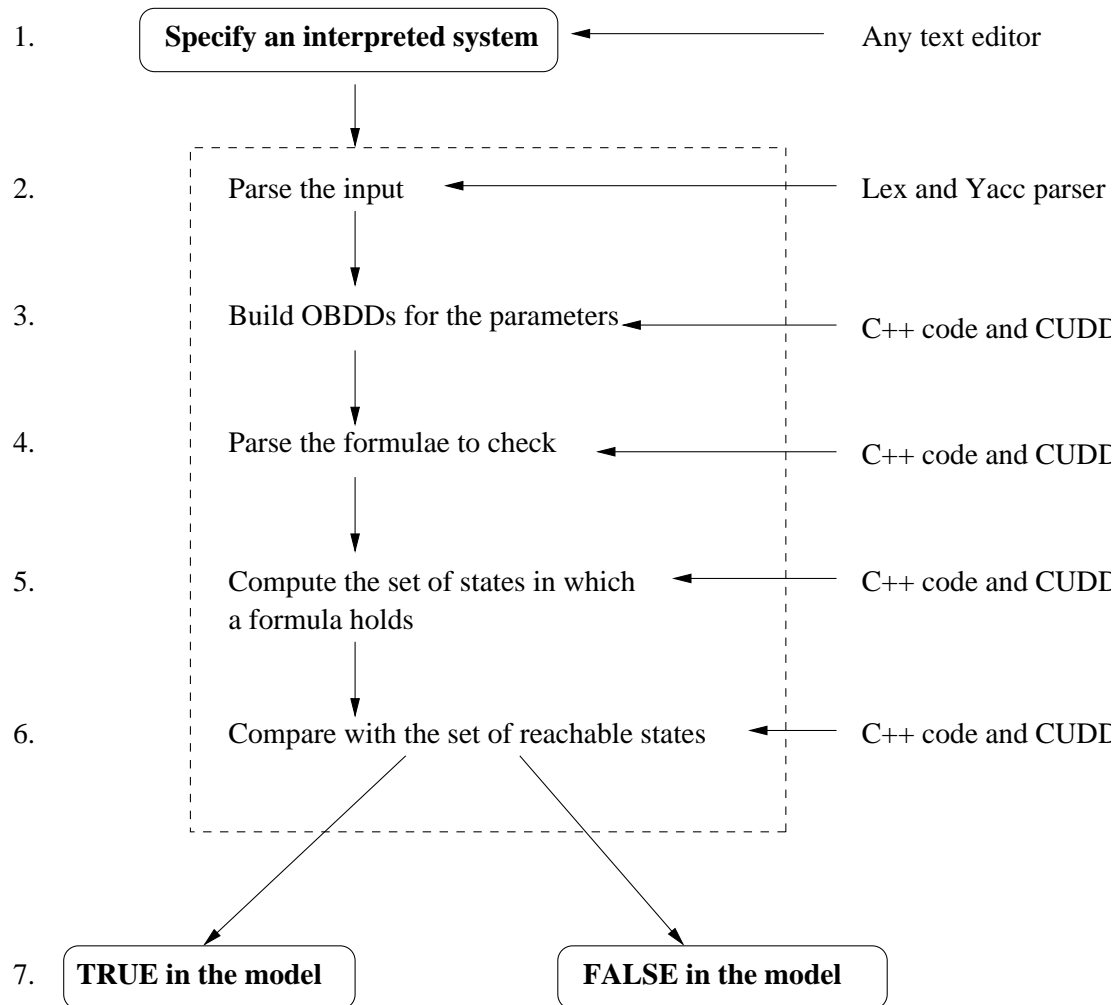
$\}$

The algorithm – 3

The set of reachable states S can be computed as the fix-point of $\tau = (I(g) \vee \exists g'(R_t(g', g) \wedge Q(g')))$ where Q denotes a set of global states. The fix-point of τ can be computed by iterating $\tau(\emptyset)$ by standard procedure.

To check whether ϕ holds in the model we can now compare $[[\phi]]$ with S .

Implementation: an overview



Example: Dining cryptographers [Ch88]

3 dining cryptographers. From vdM-Su:

Example: Dining cryptographers [Ch88]

3 dining cryptographers. From vdM-Su:

1. Each cryptographer flips a coin and observes that coin and the one at his right.

Example: Dining cryptographers [Ch88]

3 dining cryptographers. From vdM-Su:

1. Each cryptographer flips a coin and observes that coin and the one at his right.
2. If a cryptographer did not pay for the dinner he states whether the two coins he can see fell on the same side or not.

Example: Dining cryptographers [Ch88]

3 dining cryptographers. From vdM-Su:

1. Each cryptographer flips a coin and observes that coin and the one at his right.
2. If a cryptographer did not pay for the dinner he states whether the two coins he can see fell on the same side or not.
3. If a cryptographer paid for the dinner he states the opposite of what he witnessed.

Example: Dining cryptographers [Ch88]

3 dining cryptographers. From vdM-Su:

1. Each cryptographer flips a coin and observes that coin and the one at his right.
2. If a cryptographer did not pay for the dinner he states whether the two coins he can see fell on the same side or not.
3. If a cryptographer paid for the dinner he states the opposite of what he witnessed.

One can check that:

$$\neg paid_1 \implies (AX((K_1(\neg paid_1 \wedge \neg paid_2 \wedge \neg paid_3)) \vee (K_1(paid_2 \vee paid_3) \wedge \neg K_1 paid_2 \wedge \neg K_1 paid_3)))$$

Faulty cryptographers

Faulty cryptographers

Suppose now one of the agents, say agent 1, is not following the protocol, ie it may not follow its specifications. In particular: it *may say the opposite of what it should*.

Faulty cryptographers

Suppose now one of the agents, say agent 1, is not following the protocol, ie it may not follow its specifications. In particular: it *may say the opposite of what it should*.

$$\not\models ((odd \wedge \neg paid_2) \implies (K_2(paid_1 \vee paid_3)) \wedge \neg K_2 paid_1 \wedge \neg K_2 paid_3)$$

Faulty cryptographers

Suppose now one of the agents, say agent 1, is not following the protocol, ie it may not follow its specifications. In particular: it *may say the opposite of what it should*.

$$\not\models ((\text{odd} \wedge \neg \text{paid}_2) \implies (K_2(\text{paid}_1 \vee \text{paid}_3)) \wedge \neg K_2 \text{paid}_1 \wedge \neg K_2 \text{paid}_3)$$

But:

$$\models ((\text{odd} \wedge \neg \text{paid}_2) \implies ((\hat{K}_2^1(\text{paid}_1 \vee \text{paid}_3) \wedge \neg \hat{K}_2^1 \text{paid}_1 \wedge \neg \hat{K}_2^1 \text{paid}_3))$$

Demo

- IS specification syntax.
- Demo.

Experimental results - Dining Cryptos

$ M $	OBDD's nodes	Memory (MBytes)
$\approx 2.84 \cdot 10^{15}$	$\approx 10^7$	≈ 227

Table 1: Memory requirements.

Model construction	Verification	Total
89sec	7sec	136sec

Table 2: Running time (for one formula).

Experimental results - BTP with faults

$ M $	OBDD's nodes	Memory (MBytes)
$\approx 4 \cdot 10^6$	$\approx 10^3$	≈ 4.5
$\approx 7 \cdot 10^{13}$	$\approx 10^7$	≈ 220

Table 3: Memory requirements.

Model construction	Verification	Total
0.045sec	<0.01sec	0.05sec

Table 4: Running time (for one formula).

Conclusions

Conclusions

- Extension to knowledge and correct functioning behaviour of standard model checking via BDD: algorithm and implementation. The semantics is not restricted to particular classes of interpreted systems.

Conclusions

- Extension to knowledge and correct functioning behaviour of standard model checking via BDD: algorithm and implementation. The semantics is not restricted to particular classes of interpreted systems.
- Algorithm and implementation is self-contained. It treats knowledge and deontic operators as modalities, and only relies on Somenzi's libraries.

Conclusions

- Extension to knowledge and correct functioning behaviour of standard model checking via BDD: algorithm and implementation. The semantics is not restricted to particular classes of interpreted systems.
- Algorithm and implementation is self-contained. It treats knowledge and deontic operators as modalities, and only relies on Somenzi's libraries.
- One among other possibilities to check knowledge and deontic states: see SAT-based approaches.

Related and further work

Related and further work

- Improve implementation and comparison with BMC-/UMC-based tools, and other tools.

Related and further work

- Improve implementation and comparison with BMC-/UMC-based tools, and other tools.
- Comparison with Sydney.