Building a Symbolic Model Checker from Formal language Description

$\Sigma DD$ and StrataGEM

Didier Buchs and Edmundo Lopez

Geneva University

5 mars 2015
Motivations

- Difficult to build your own symbolic model checker
- Hard to reuse existing work
  - Semantic construction
  - Optimisation
  - Decision Diagram encoding

\[ M \models \Phi \iff DDCompute_\Phi(Enc_{DD}(M)) \]
Remark:

- SAT more popular i.e. modular and based on propositional logic:

\[ M \models \Phi \iff \text{SatCompute}(\text{Enc}_{\text{prop}}(\Phi) \land \text{Enc}_{\text{prop}}(M)) \]
Observation:

- Large semantic gap between analysed language and DD
- Decision Diagram based on set of items:
  \[
  \text{Enc}: \emptyset(\text{States}) \rightarrow DD
  \]
  \[
  \text{Enc}(s_1 \cup s_2) = \text{Enc}(s_1) \cup_{DD} \text{Enc}(s_2)
  \]
- Can we describe them state by state?
- Can we extend the computations to state efficiently?

\[
M \models \Phi \iff DD\text{Compute}(\text{Enc}_{DD}(\text{RewTr}(\Phi)) \circ \text{Enc}_{DD}(\text{RewTr}(M)))
\]
Introduction : Topics

- Points to address
  - How to express Semantics?
  - What Model Checking technique?
  - How to express Computations?
Introduction : Topics

• Points to address
  • How to express Semantics?
  • What Model Checking technique?
  • How to express Computations?

• Formal Basis
  • $\Sigma DD$
  • Term Rewriting
  • Strategies
Introduction : Global view

- Formalism
  - Abstract Semantics (SOS Rules)
    - User defined translation
  - Set rewriting (Strategies)
  - Automated translation
  - Symbolic Structures (Decision Diagrams)

Our approach

This Presentation
Credits

- Prof invité (2007) at LIP6,
  - SDD : Jean-Michel Couvreur and Yann Thierry-Mieg
  - Operations : Alexandre Hamez and Alban Linard

- Collaboration

- Work done at SMV, University of Geneva
  - ΣDD (2009) : Steve Hostettler and Edmundo Lopez
  - Alpina (2012) : Steve Hostettler and Alexis Marechal
Terms

- A signature $\Sigma = \langle S, Op \rangle$.
  
  $S = \{\text{bool}, \text{nat}, \text{list}\}$

  $Op = \{ 0 : \rightarrow \text{nat};$
  $s : \text{nat} \rightarrow \text{nat};$
  $+ : \text{nat}, \text{nat} \rightarrow \text{nat}; \}$

- Inductively defined terms: $T_{\Sigma}$
  $0 + s(s(0))$

- Inductively defined terms with variables: $T_{\Sigma}(X)$
  $0 + s(s(x))$
Encoding : A ’n’ digit counter

Signature
null : → counter;
digit : nat10, counter → counter;

Terms :

digit(d₃, digit(d₂, digit(d₁, null)))

digit(s(s(0)), digit(s(0), digit(0, null)))

"2 1 0"
Rewriting

Rewrite rule: \( t_l, t_r \in T_\Sigma(X) : t_l \rightsquigarrow t_r \)

Example (functional rules):
Rule 1: \( +(0, x) \rightsquigarrow x \)
Rule 2: \( +(s(x), y) \rightsquigarrow s(+ (x, y)) \)

rewriting as computation of semantics
\( +(s(0), s(0)) \rightsquigarrow s(+ (0, s(0))) \rightsquigarrow s(s(0))) \)
Rewriting for states

Example (partial/basic rules):

\[\text{digit}(X, C) \leadsto \text{digit}(s(X), C)\]

\[\text{digit}(X, \text{digit}(s(s(s(s(s(s(0))))))))), C)) \leadsto \text{digit}(s(X), \text{digit}(0, C))\]

What about combining these rules?

Semantics defined on basic rewriting and strategies:

\[\text{Reach}_M(s_0) = \left\{ s' \mid s_0 \leadsto . \leadsto .s' \right\} = \{ s_1, s_2, \ldots, s_n \} \]
Set of terms

We propose to consider set of terms: \( s = \{ t_1, t_2, \ldots, t_n \} \)

\[
Rew(\{t_1, t_2, \ldots, t_n\}) = \bigcup_{t_i} Rew(t_i)
\]

- Different (choice) strategies on rewriting of confluent and terminating systems produce similar results \( Rew_{strat}(s) = Rew_{strat'}(s) \).
In $\Sigma DD$ a structure represents a set of terms.

$$\sigma \in \text{SIGDD}_\Sigma , \sigma = \text{enc}(\{t_1, t_2, \ldots, t_n\}) \text{ where } t_i \in T_\Sigma$$

$$\sigma \in \text{SIGDD}_\Sigma , \text{dec}(\sigma) = \{t_1, t_2, \ldots, t_n\} \text{ where } t_i \in T_\Sigma$$

Encoding and decoding $inc$ and $dec$ are homomorphisms.

$$\forall \sigma \in \text{SIGDD}_\Sigma , \sigma = \text{enc}(\text{dec}(\sigma))$$

$$\forall t_i \in T_\Sigma , \{t_1, t_2, \ldots, t_n\} = \text{dec}(\text{enc}(\{t_1, t_2, \ldots, t_n\}))$$

Perform rewriting on $\Sigma DD$ :

$$\text{Rew}(s) = \text{dec}(\text{Rew}_{\Sigma DD}(\text{enc}(s)))$$
Set of terms

\[ \{ + (0, s(0)), + (s(0), s(0)) \} \]
Set of terms

\[ \{ + (0, s(0)), +(s(0), s(0)) \} \]
Normal Form

Rule 1 : \((0, x) \leadsto x\)
Rule 2 : \((s(x), y) \leadsto s(+(x, y))\)
\(\{s(0), s(s(0))\}\)

Didier Buchs and Edmundo Lopez
Building a Symbolic Model Checker from Formal language Description/5 mars 2015
More sharing on set of terms

\{ + (0, + (0, s(0))), + (s(s(0), + (0, s(0))), + (0, + (s(0), s(0))), + (s(s(0), + (s(0), s(0)))) \}
Sharing/Rewriting on set of terms

Normal form: \{ s(0), s(s(0)), s(s(s(0))), s(s(s(s(0)))) \}

Rewrite of several terms in one step!
Complete Atomic Boolean Algebra (CABA). A complete Boolean Algebra is a (complete distributive lattice)
\[ \langle L, \lor, \land, 0, 1 \rangle \]
equipped with a unary \textit{complementation} operation \( \neg \), satisfying \( a \lor \neg a = 1 \) and \( a \land \neg a = 0 \) for all \( a \in L \).
Encoding Relation

Definition (Encoding Relation)

The binary relation \( R = \langle A, B, G \rangle \) is encoded by \( R' = \langle A', B', G' \rangle \), where \( A' \subseteq \mathcal{P}(A) \) and \( B' \subseteq \mathcal{P}(B) \), if and only if one of the following holds:

- \( G = \emptyset \) and \( G' = \{(A, \emptyset)\} \),
- \( (x, y) \in G \iff (X, Y) \in G' \) with \( x \in X \) and \( y \in Y \)
**Encoding Relation : example**

\[ G = \{(1, 1), (2, 1), (2, 2), (3, 1), (3, 2), (3, 3), (4, 1), (4, 2), (4, 3)\} \]

we exhibit the encoding:

\[
\begin{align*}
A' &= \begin{cases} 
\{1\}, & \{2\}, & \{3, 4\} 
\end{cases} \\
B' &= \begin{cases} 
\{1\}, & \{1, 2\}, & \{1, 2, 3\} 
\end{cases} \\
G' &= \begin{cases} 
(\{1\}, \{1\}), & (\{2\}, \{1, 2\}), & (\{3, 4\}, \{1, 2, 3\}) 
\end{cases}
\end{align*}
\]
Injective partitionned functions (IPF)

The set of IPF between $A$ and $B$, noted $\Delta(A, B)$, is defined as follows:

$$\Delta(A, B) = \{ f : \pi_f \to \mathcal{P}(B) \setminus \emptyset_B \mid \pi_f \subset \mathcal{P}(A) \setminus \emptyset_A \text{ and } \forall X, Y \in \pi_f : X \neq Y \implies X \wedge Y = \emptyset_A \text{ and } f(X) \neq f(Y) \}$$

$$\cup \{ 1_A \mapsto 0_B \}$$
The CABA structure of $\mathcal{B}(A, B)$

$\Delta(A, B)$ is CABA.

- $\cup$, $\cap$ on $\Delta(A, B)$
- $\neg$ on $\Delta(A, B)$
n-ary relation: currying (IIPF)

As example, we define the ternary relation \( \text{the-sum-is-pair} = \langle A, B, C, G \rangle \), with \( A = \{1, 2, 3, 4\} \), \( B = \{1, 2, 3\} \), \( C = \{1, 2\} \) and

\[
G = \{(1, 1, 2), (1, 2, 1), (1, 3, 2), (2, 1, 1), (2, 2, 2), (2, 3, 1), (3, 1, 2), (3, 2, 1), (3, 3, 2), (4, 1, 1), (4, 2, 2), (4, 3, 1)\}
\]

We can encode this relation in an IPF \( f \in \Delta_{A,B,C} \):

\[
f : \begin{cases} 
\{1, 3\} &\mapsto f_1 \\
\{2, 4\} &\mapsto f_2
\end{cases}
\]

\[
f_1 : \begin{cases} 
\{1, 3\} &\mapsto g_2 \\
\{2\} &\mapsto g_1
\end{cases}
\]

\[
f_2 : \begin{cases} 
\{1, 3\} &\mapsto g_1 \\
\{2\} &\mapsto g_2
\end{cases}
\]

\[
g_1 : \{1\} \mapsto 1 \\
g_2 : \{2\} \mapsto 1
\]
Definition ($\Sigma \text{DD}$)

Let $\Sigma = \langle S, F \rangle$ and $X$ be a set of variables. The set of $\Sigma \text{DD}$ over $\Sigma$ and $X$ consists of a family $(\Sigma \text{DD}^{\Sigma,X}_s)_{s \in S}$, where each $\Sigma \text{DD}^{\Sigma,X}_s$ is limit of the sequence defined as:

- $\Sigma \text{DD}^0_s = \Delta F_{\epsilon,s} \cup X_s$
- $\Sigma \text{DD}^{n+1}_s = \Sigma \text{DD}^n_s \cup \bigcup_{F_{s_1 \ldots s_k}, s \in F} \Delta (F_{s_1 \ldots s_k}, s \Delta \Sigma \text{DD}^n_{s_1}, \ldots, \Sigma \text{DD}^n_{s_k})$
Establish links between Rewriting techniques and operations on decision diagrams. We would have **performance** in mind.
Reminder on Rewriting a la TOM

Based on elementary rewrite rules, we can apply on terms a basic rewrite step.

\[ \text{Rew}_{Ax}[t] = \ldots \]

\[
\exists \sigma, \\
(\sigma(l) = t) \Rightarrow \text{Rew}_{Ax \cup \{<l,r>\}}[t] = \sigma(r)
\]
Reminder on Strategies

Way to find the context of a rewriting step!

\[ Strat(S) : (T_\Sigma \cup \{\text{fail}\}) \rightarrow (T_\Sigma \cup \{\text{fail}\}) \]

More generally:

\[ Strat(S) : (\varnothing(T_\Sigma) \cup \{\text{fail}\}) \rightarrow \varnothing(T_\Sigma) \cup \{\text{fail}\} \]

If \( Strat(s) \) is defined, terms \( t \) will be rewritten with:

\[ Strat(Rew_{Ax})[t] \]

Obviously:

\[ (S)[\text{fail}] = \text{fail} \]
Reminder on Strategies:

Basic operations 1 (TOM)

\[
\begin{align*}
(\text{Identity})[t] &= t \\
(\text{Fail})[t] &= \text{fail} \\
(\text{Sequence}(s1, s2))[t] &= \text{fail} \iff (s1)[t] = \text{fail} \\
(\text{Sequence}(s1, s2))[t] &= (s2)[t'] \iff (s1)[t] = t'
\end{align*}
\]

\[
\begin{align*}
(\text{Choice}(s1, s2))[t] &= t' \iff (s1)[t] = t' \\
(\text{Choice}(s1, s2))[t] &= (s2)[t] \iff (s1)[t] = \text{fail}
\end{align*}
\]
Strategies on sets

Natural extension

\[ S[\{t_1, \ldots, t_n\}] = \{S[t_1], \ldots, S[t_n]\} \]

Set strategies

\[ \text{Union}(S_1, S_2)[T] = S_1[T] \cup S_2[T], \text{ if both succeed} \]

\[ \text{Fixpoint}(S)[T] = \mu T. S[T] \]
Restrictions

terminating

\[ x \sim s(x) \]
\[ s(x) \sim + (x, y) \]
\[ + (x, y) \sim + (y, x) \]

linear

\[ + (x, x) \sim x \]
\[ + (x, y) \sim + (x, x) \]

no-condition

\[ x > y \Rightarrow s(x) - s(y) = x - y \]
Example of strategies

Innermost Evaluation:

\[ \text{Try}(S) = \text{Choice}(S, \text{Identity}) \]

\[ \text{Innermost}(S) = \mu x. \text{Sequence}(\text{All}(x), \text{Try}(\text{Sequence}(S, x))) \]
**Computation on $\Sigma DD$**

- $\Sigma DD$ employs homomorphisms (set regularity) for implementing rewriting, $Rew_{\Sigma DD} \in Hom$.
- These homomorphisms can be defined for strategies: $Rew_{strat,\Sigma DD}$.
- On terminating and confluent systems $\Sigma DD$, rewriting respects sets: $Rew_{strat,\Sigma DD} \in Hom$ for deterministic $strat$ strategies.

Some strategies are better (performance) than others as in rewriting and similarly in decision diagrams.
IPF can be defined with different representation (automaton, pressburger arithmetic,...), so do $\Sigma$DD...
IPF can be defined with different representation (automaton, pressburger arithmetic,...), so do $\Sigma DD$

can we compose Rew, ... easily? by strategies?
Conclusion

- IPF can be defined with different representation (automaton, pressburger arithmetic,...), so do $\Sigma$DD
- Can we compose Rew, ... easily? by strategies?
- Can we define Design Patterns (Edmundo’s talk)?
Conclusion

- IPF can be defined with different representation (automaton, pressburger arithmetic,...), so do \( \Sigma DD \)
- can we compose Rew, ... easily? by strategies?
- Can we define Design Patterns (Edmundo’s talk)?
- 

---

Didier Buchs and Edmundo Lopez  Building a Symbolic Model Checker from Formal language Description/5 mars 2015
Thank You for your attention!