Building a Symbolic Model Checker from Formal language Description

ΣDD and StrataGEM

Didier Buchs and Edmundo Lopez

Geneva University

5 mars 2015

Introduction : Context

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

$$M \models \Phi \Leftrightarrow DDCompute_{\Phi}(Enc_{DD}(M))$$

Introduction : Context

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

$$M \models \Phi \Leftrightarrow SatCompute(Enc_{prop}(\Phi) \wedge Enc_{prop}(M))$$

Introduction : Context

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

$$M \models \Phi \Leftrightarrow DDCompute(Enc_{DD}(RewTr(\Phi)) \circ Enc_{DD}(RewTr(M)))$$

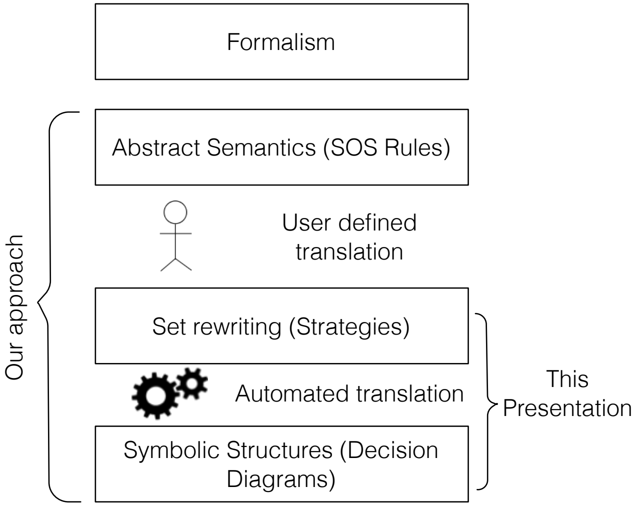
Introduction : Topics

- 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
  - ΣDD
  - Term Rewriting
  - Strategies

Introduction : Global view



Credits

- Prof invité (2007) at LIP6,
  - SDD : Jean-Michel Couvreur and Yann Thierry-Mieg
  - Operations : Alexandre Hamez and Alban Linard
- Collaboration
  - *PolyDD* (2010) : Alban Linard, Emmanuel Paviot-Adet and Fabrice Kordon.
- Work done at SMV, University of Geneva
  - ΣDD (2009) : Steve Hostettler and Edmundo Lopez
  - Alpina (2012) : Steve Hostettler and Alexis Marechal

Terms

- A signature $\Sigma =< S, Op >$.
  $S = \{bool, nat, list\}$
  Op= $\{\ 0 : \rightarrow nat;$
  s : $nat \rightarrow nat;$
  $+ : nat, nat \rightarrow nat; \}$
- Inductively defined terms : T_{Σ}
  $0 + s(s(0))$
- Inductively defined terms with variables : $T_{\Sigma}(X)$
  $0 + s(s(x))$

Encoding : A 'n' digit counter

Signature

null : $\rightarrow$ *counter*;

digit : $nat10, counter \rightarrow counter$;

Terms :

$$digit(d_3, digit(d_2, digit(d_1, null)))$$

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

"2 1 0"

Rewriting

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

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

rewriting as computation of semantics
$+(s(0), s(0)) \leadsto s(+(0, s(0))) \leadsto s(s(0)))$

Rewriting for states

Example(partial/basic rules) :

$$digit(X, C) \rightsquigarrow digit(s(X), C)$$

$$digit(X, digit(s(s(s(s(s(s(s(s(s(0))))))))), C)) \rightsquigarrow digit(s(X), digit(0, C))$$

What about combining these rules ?

Semantics defined on basic rewriting and strategies :
$Reach_M(s_0) = \{s' | s_0 \rightsquigarrow . \rightsquigarrow .s'\} = \{s_1, s_2, \ldots, s_n\}$

Set of terms

We propose to consider set of terms : $s = \{t_1, t_2, ..., t_n\}$

$$Rew(\{t_1, t_2, ..., t_n\}) = \bigcup_{t_i} Rew(t_i)$$

- Different (choice) strategies on rewriting of confluent and terminating systems produce similar results $Rew_{strat}(s) = Rew_{strat'}(s)$.

ΣDD

In ΣDD a structure represents a set of terms.

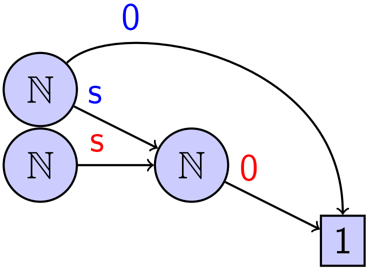
$\sigma \in \mathbb{SIGDD}_\Sigma$, $\sigma = enc(\{t_1, t_2, ..., t_n\})$ where $t_i \in T_\Sigma$
$\sigma \in \mathbb{SIGDD}_\Sigma$, $dec(\sigma) = \{t_1, t_2, ..., t_n\}$ where $t_i \in T_\Sigma$
Encoding and decoding *inc* and *dec* are homomorphisms.

$\forall \sigma \in \mathbb{SIGDD}_\Sigma$, $\sigma = enc(dec(\sigma))$
$\forall t_i \in T_\Sigma, \ \{t_1, t_2, ..., t_n\} = dec(enc(\{t_1, t_2, ..., t_n\}))$

Perform rewriting on ΣDD :
$Rew(s) = dec(Rew_{\Sigma DD}(enc(s)))$

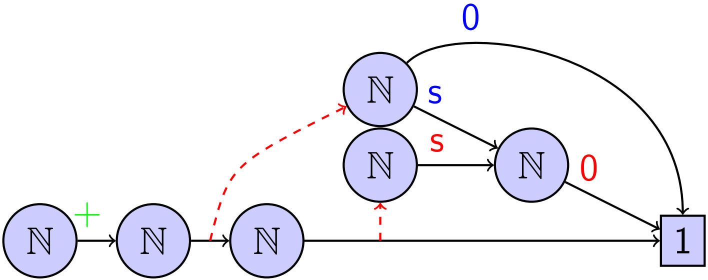
Set of terms

$\{+(0,s(0)),+(s(0),s(0))\}$



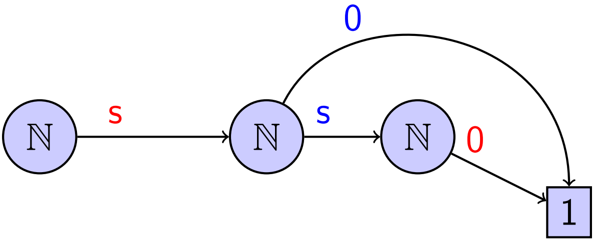
Set of terms

{+(0,s(0)),+(s(0),s(0))}



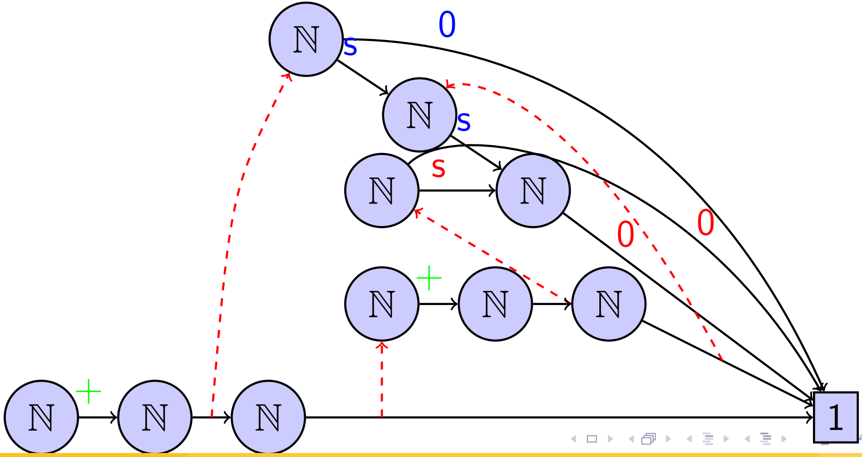
Normal Form

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



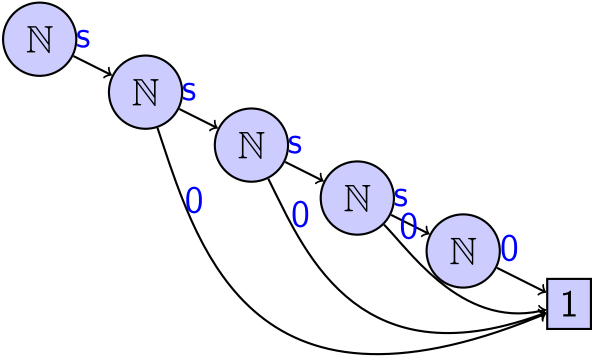
More sharing on set of terms

{+(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 !



ΣDD structure

Complete Atomic Boolean Algebra (CABA).
A complete Boolean Algebra is a (complete distributive lattice)
$\langle L, \vee, \wedge, \boxed{0}, \boxed{1} \rangle$
equipped with a unary *complementation* operation $\neg$,
satisfying $a \vee \neg a = \boxed{1}$ and $a \wedge \neg a = \boxed{0}$ 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 :

- $G = \varnothing$ and $G' = \{(A, \varnothing)\}$,
- $(x, y) \in G \Leftrightarrow (X, Y) \in G'$ 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 :

$$
\begin{array}{lcl}
A' & = & \{ \quad \{1\}, \quad \{2\}, \quad \{3,4\} \ \} \\
B' & = & \{ \quad \{1\}, \quad \{1,2\}, \quad \{1,2,3\} \ \} \\
G' & = & \{ \ (\{1\},\{1\}), \ (\{2\},\{1,2\}), \ (\{3,4\},\{1,2,3\}) \ \}
\end{array}
$$

Injective partitionned functions (IPF)

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

$$\Delta(A,B) = \{\ f : \pi_f \to \mathcal{P}(B) \backslash \boxed{0}_B \mid \pi_f \subset \mathcal{P}(A) \backslash \boxed{0}_A \text{ and } \forall X, Y \in \pi_f : X \neq Y \implies X \wedge Y = \boxed{0}_A \text{ and } f(X) \neq f(Y)\} \cup \{\boxed{1}_A \mapsto \boxed{0}_B\}$$

IPF as CABA

The CABA structure of $\mathcal{B}(A, B)$

$\Longrightarrow$

$\Delta(A, B)$ 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 *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 $f \in \Delta_{A,B,C}$:

$$f : \begin{cases} \{1,3\} & \mapsto & f_1 \\ \{2,4\} & \mapsto & f_2 \end{cases} \quad f_1 : \begin{cases} \{1,3\} & \mapsto & g_2 \\ \{2\} & \mapsto & g_1 \end{cases} \quad f_2 : \begin{cases} \{1,3\} & \mapsto & g_1 \\ \{2\} & \mapsto & g_2 \end{cases}$$

$$g_1 : \{ \ \{1\} \quad \mapsto \quad \boxed{1} \qquad g_2 : \{ \ \{2\} \quad \mapsto \quad \boxed{1}$$

ΣDD

Definition (ΣDD)

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

- $\Sigma\mathbb{DD}_s^0 \quad = \Delta_{F_{\epsilon,s} \cup X_s}$
- $\Sigma\mathbb{DD}_s^{n+1} =$
  $\Sigma\mathbb{DD}_s^n \cup \biguplus_{F_{s_1 \ldots s_k, s} \in F} \Delta(F_{s_1 \ldots s_k, s}, \Delta_{\Sigma\mathbb{DD}_{s_1}^n, \ldots, \Sigma\mathbb{DD}_{s_k}^n})$

Aim of the rest of this presentation

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.
$Rew_{Ax}[t] = \ldots$

$\exists \sigma,$
$(\sigma(l) = t) \Rightarrow Rew_{Ax \cup \{<l,r>\}}[t] = \sigma(r)$

Reminder on Strategies

Way to find the context of a rewriting step !

$$Strat(S) : (T_\Sigma \cup \{fail\}) \to (T_\Sigma \cup \{fail\})$$

More generally :

$$Strat(S) : (\wp(T_\Sigma) \cup \{fail\}) \to \wp(T_\Sigma) \cup \{fail\}$$

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

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

Obviously :

$$(S)[fail] = fail$$

Reminder on Strategies : Basic operations 1 (TOM)

$$
\begin{aligned}
(Identity)[t] &= t \\
(Fail)[t] &= fail \\
(Sequence(s1, s2))[t] = fail \Leftarrow (s1)[t] &= fail \\
(Sequence(s1, s2))[t] = (s2)[t'] \Leftarrow (s1)[t] &= t'
\end{aligned}
$$

$$
\begin{aligned}
(Choice(s1, s2))[t] = t' \Leftarrow (s1)[t] &= t' \\
(Choice(s1, s2))[t] = (s2)[t] \Leftarrow (s1)[t] &= fail
\end{aligned}
$$

Strategies on sets

Natural extension

$$S[\{t1, , tn\}] = \{S[t1], , S[tn]\}$$

Set strategies

$$Union(S1, S2)[T] = S1[T] \cup S2[T], \text{ if both succeed}$$

$$Fixpoint(S)[T] = \mu T.S[T]$$

Restrictions

terminating

$$x \rightsquigarrow s(x)$$

$$s(x) \rightsquigarrow +(x, y)$$

$$+(x, y) \rightsquigarrow +(y, x)$$

linear

$$+(x, x) \rightsquigarrow x$$

$$+(x, y) \rightsquigarrow +(x, x)$$

no-condition

$$x > y \Rightarrow s(x) - s(y) = x - y$$

Example of strategies

Innermost Evaluation :

$$Try(S) = Choice(S, Identity)$$

$$Innermost(S) = \mu x.Sequence(All(x), Try(Sequence(S, x)))$$

Computation on ΣDD

- ΣDD 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 ΣDD 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.

Conclusion

- IPF can be defined with different representation (automaton, pressburger arithmetic,...), so do $\Sigma\mathrm{DD}$

Conclusion

- IPF can be defined with different representation (automaton, pressburger arithmetic,...), so do $\Sigma\mathrm{DD}$
- can we compose Rew, ... easily ? by strategies ?

Conclusion

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

Thank You for your attention !