Model Checking, Symbolic approaches and Set Family Decision Diagrams

Dimitri Racordon, Damien Morard, Steve Hostettler
Didier Buchs

Centre Universitaire d'Informatique, Université de Genève

December 2, 2021

What are the barrier for using model checking

Systems are often very complex

Formal languages are . . . formal

The infamous state space explosion problem [11]

trop d'états

DSL:

domain specific language

DSL:

???:

The State Space Explosion (Example)

Dining philosopher problem

1 byte per state

200 philosophers $\Longrightarrow$ $2.5x10^{125}$ states (# of atoms in the observable universe: 10^{80})

28421709400000000000000000000000000000000000000
00000000000000000000000000000000000000000000000
000000000000000000 TBytes

The State Space Explosion (Solutions)

- Reduce the search space
  - Partial orders (Model checking by representatives)
  - Abstractions
  - Symmetry based (Quotient graphs)

The State Space Explosion (Solutions)

- Reduce the search space
  - Partial orders (Model checking by representatives)
  - Abstractions
  - Symmetry based (Quotient graphs)
- Improve the computation and storing power bof

The State Space Explosion (Solutions)

- Reduce the search space
  - Partial orders (Model checking by representatives)
  - Abstractions
  - Symmetry based (Quotient graphs)
- Improve the computation and storing power
- Better state space representation
  - Symbolic approaches

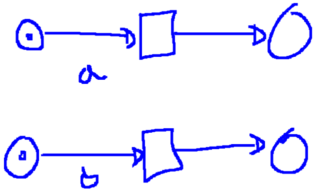
The State Space Explosion (Solutions)

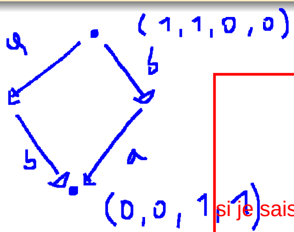
- Reduce the search space
  - Partial orders (Model checking by representatives)
  - Abstractions
  - Symmetry based (Quotient graphs)
- Improve the computation and storing power bof
- Better state space representation
  - Symbolic approaches
- Better satisfaction discovery
  - SAT/SMT solvers

Reducing the search space: Partial Orders (1)

- Avoid storing and checking paths that are considered equivalent w.r.t the property to check.

- Avoid storing and checking paths that are considered equivalent w.r.t the property to check.
- Exploit the commutativity of concurrently executed transitions, which result in the same state when executed in different orders (diamond property) [5, 6].



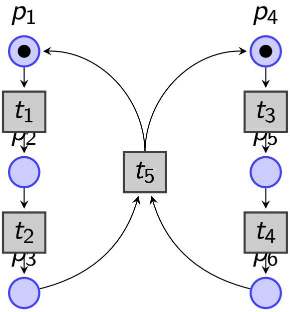


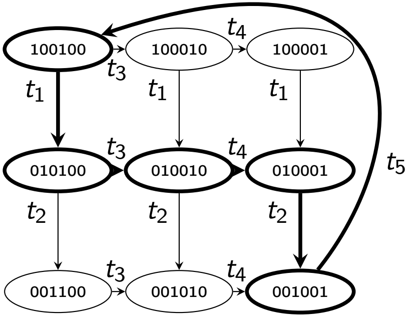
si je sais que a et b so

Reducing the search space: Partial Orders (1)

- Avoid storing and checking paths that are considered equivalent w.r.t the property to check.
- Exploit the commutativity of concurrently executed transitions, which result in the same state when executed in different orders (diamond property) [5, 6].
- Different techniques (ample sets [9], stubborn sets [10])

Reducing the search space: Partial Orders (2)





Reducing the search space: Abstractions

- Big models

Reducing the search space: Abstractions

- Big models
- Make a reduced model according to what one wants to check, while preserving interesting properties

Reducing the search space: Abstractions

- Big models
- Make a reduced model according to what one wants to check, while preserving interesting properties
- Check the counter-example against the complete model and refine the abstraction accordingly

Reducing the search space: Abstractions

- Big models
- Make a reduced model according to what one wants to check, while preserving interesting properties
- Check the counter-example against the complete model and refine the abstraction accordingly
- CEGAR approach [7, 4] (Counter Example Guided Abstraction Refinement)

Reducing the search space: CEGAR

To verify $C \models AGp$ do

1. build finite Kripke structure $A \rhd C$ an abstraction (homomorphic),
2. model-check $\boxed{A \models AGp}$,
3. if this holds then report $C \models AGp$ and stop,
4. otherwise validate the counterexample on C, i.e., find a corresponding concrete counterexample,
5. if a corresponding concrete counterexample exists then report $C \not\models AGp$ and stop,
6. otherwise use the spurious counterexample to refine A and restart from 2.

mauvaise abstraction

Reducing the search space: Quotient graphs (1)

- Exploit symmetries between the states

Reducing the search space: Quotient graphs (1)

- Exploit symmetries between the states
- Define equivalence relation on the state space

Reducing the search space: Quotient graphs (1)

- Exploit symmetries between the states
- Define equivalence relation on the state space
- Bisimulation equivalent to the original model

Reducing the search space: Quotient graphs (1)

- Exploit symmetries between the states
- Define equivalence relation on the state space
- Bisimulation equivalent to the original model
- Efficiency is highly dependent on the system (exponential at best, no reduction at worst)

Reducing the search space: Quotient graphs (2)

Example

- Let's consider a system in which two clients (c_1 and c_2) communicate with a server.

Reducing the search space: Quotient graphs (2)

Example

- Let's consider a system in which two clients (c_1 and c_2) communicate with a server.
- c_1 and c_2 have an identical behaviour.

Reducing the search space: Quotient graphs (2)

Example

- Let's consider a system in which two clients (c_1 and c_2) communicate with a server.
- c_1 and c_2 have an identical behaviour.
- Instead of considering the state s_1 in which the client c_1 sends a message to the server and another state in which the client c_2 does the same, we consider the equivalent state s' in which one of the client sends a message.

Better state space representation: Symbolism

Two approaches:

Better state space representation: Symbolism

Two approaches:

Representation of symbolic states

Define equivalence classes between states and perform check on the classes

Better state space representation: Symbolism

Two approaches:

Representation of symbolic states

Define equivalence classes between states and perform check on the classes

Symbolic representation of the states

Efficient representation of all the states using dedicated data structures (BDD [1], MDD [2], DDD [3]. . .) based on their similarities [1]

Decision Diagrams

How to compute efficiently on sets ?

- represent sets in a compact way
- compute on a whole set instead on a single element
  - aka SIMD or *graphic card computing*
- respect union : set homomorphism

various approaches based on decision diagrams. See [8] for a generalization.

Transformation

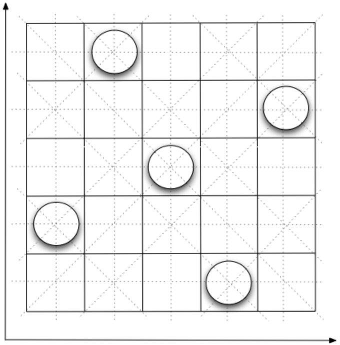
The purpose of a transformation is to translate data in a more convenient space to perform operations.
Existing transformations:

- Fourier Transform, Laplace transform ... related to spectral analysis on continuous domain
- Decision diagrams

$$\begin{tikzcd}
V \in Set \arrow[r] \arrow[d, dashed] & tr(V) \in SFDD \arrow[d] \\
F(V) \in Set & H_F(tr(V)) \in SFDD \arrow[l]
\end{tikzcd}$$

Solving problems

- Define the problem as a set of boolean equations.
- Build a ROBDD of the equation and then check if there is a solution.
- The N-Queens problem : How to put n queens on a standard n x n chess board?



Solving problems (cont'd)

Let $C_{i,j}$ be the constraint for the position i,j and $Q_{i,j}$ the presence of a queen at position i,j:

$$C1_{i,j} = Q_{i,j}. \bigwedge_{1 \leq k \leq n, k \neq i} \neg Q_{k,j}. \neg Q_{k,j+i-k}. \neg Q_{k,j+k-i}. \bigwedge_{1 \leq l \leq n, l \neq j} \neg Q_{i,l}$$

No other queens on diagonals and line and columns.
$C_{i,j}$ must be satisfied for all i and j.

$$C_{i,j} = C1_{i,j}. \bigwedge_{1 \leq l \leq n} (\bigvee_{1 \leq k \leq n} C1_{k,l})$$

At least one queen per column, to avoid no queens at all as a solution.

Solving problems (cont'd)

N	Variables	Nodes	ROBDD size	ROBDD/log Nodes
4	16	2^{16}	30	6.3
5	25	2^{25}	195	26
6	36	2^{36}	133	12.3
7	49	2^{49}	1449	103
8	64	2^{64}	3887	201

This solution scale-up well compared to other algorithms. Look , for instance, at genetic algorithms http://www.iba.k.u-tokyo.ac.jp/english/userlog.cgi?queenrun

A SFDD is a directed acyclic graph where

- each node represent a term
- each node has two children, indicating whether or not the term is contained
- each path from the root to an accepting terminal represents a set of terms
- terms are totally ordered

enc({ {a,b}, {c,d}, {a,c}, ∅ }) ⇒ . . .

basée sur les éléments {a,b,c,d}

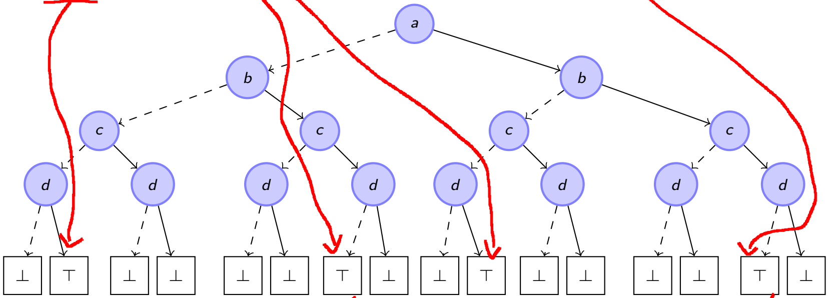
a < b < c < d

Encodes with the order $a < b < c < d$ the sets:

- $\{a, b, c\}$
- $\{a, d\}$
- $\{b, c\}$
- $\{d\}$

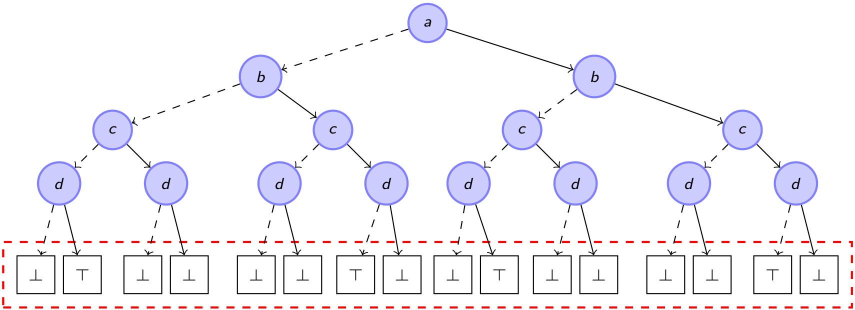
$\top$ accepté

$\perp$ rejeté



Encodes with the order $a < b < c < d$ the sets:

- $\{a, b, c\}$
- $\{a, d\}$
- $\{b, c\}$
- $\{d\}$

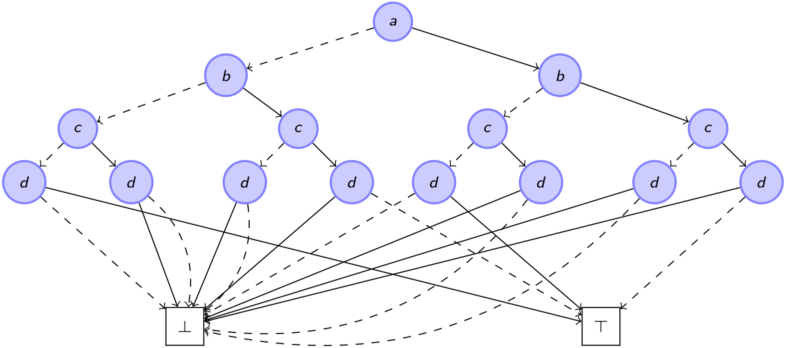


Encodes with the order $a < b < c < d$ the sets:

- $\{a, b, c\}$
- $\{a, d\}$
- $\{b, c\}$
- $\{d\}$

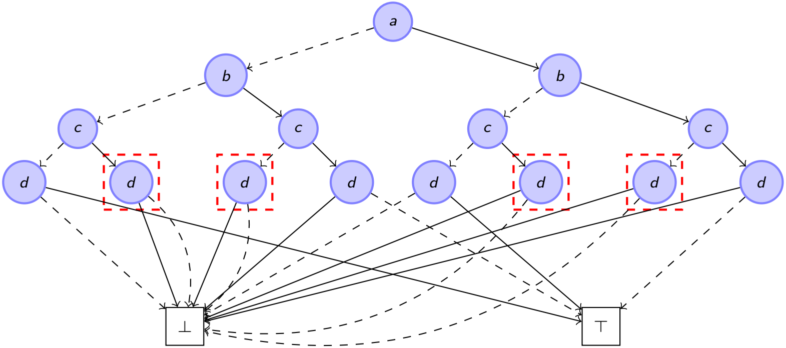
on va pas dupliquer ce qui est identi

c'est le 1er principe



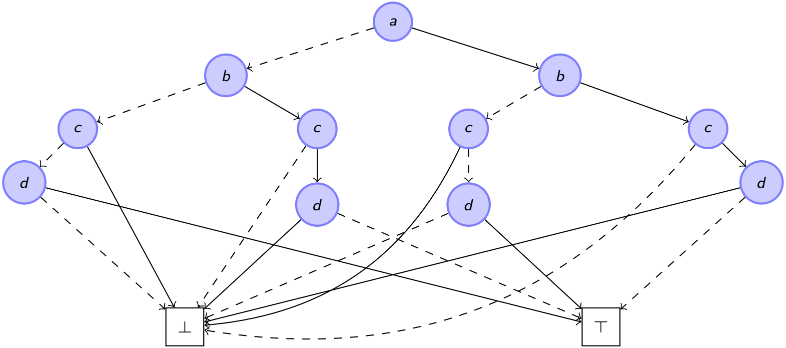
Encodes with the order $a < b < c < d$ the sets:

- $\{a, b, c\}$
- $\{a, d\}$
- $\{b, c\}$
- $\{d\}$



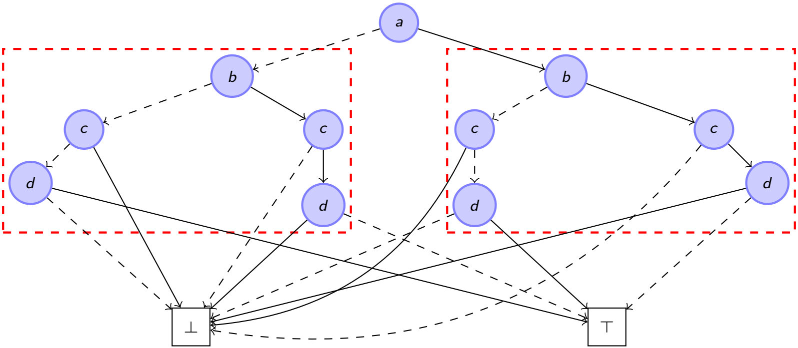
Encodes with the order $a < b < c < d$ the sets:

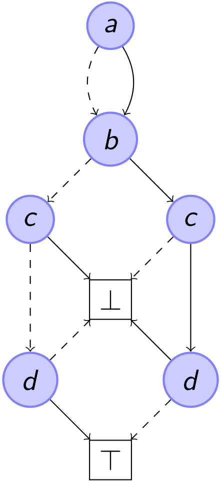
- $\{a, b, c\}$
- $\{a, d\}$
- $\{b, c\}$
- $\{d\}$



Encodes with the order $a < b < c < d$ the sets:

- $\{a, b, c\}$
- $\{a, d\}$
- $\{b, c\}$
- $\{d\}$

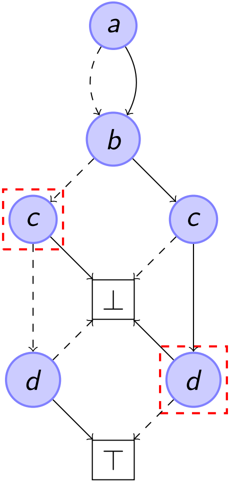




Encodes with the order $a < b < c < d$ the sets:

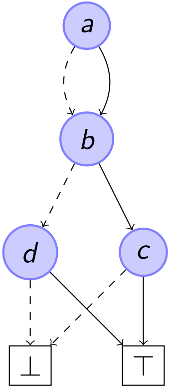
- $\{a, b, c\}$
- $\{a, d\}$
- $\{b, c\}$
- $\{d\}$

Example Reduction Part 4: Remove takes nodes whose then branch is $\perp$



Encodes with the order $a < b < c < d$ the sets:

- $\{a, b, c\}$
- $\{a, d\}$
- $\{b, c\}$
- $\{d\}$



Encodes with the order $a < b < c < d$ the sets:

- $\{a, b, c\}$
- $\{a, d\}$
- $\{b, c\}$
- $\{d\}$

Shannon decomposition

Given the family of set:

$$\{\{a, b, c\}, \{a, d\}, \{b, c\}, \{d\}\}$$

We can prefix these sets by the presence of a and the absence of a noted $\overline{a}$

$$\{a\} \otimes \{\{b, c\}, \{d\}\} \cup \{\overline{a}\} \otimes \{\{b, c\}, \{d\}\}$$

The right factor is similar and then be shared:

$$\{a, \overline{a}\} \otimes \{\{b, c\}, \{d\}\}$$

We continue the process on b and $\overline{b}$

$$\{a, \overline{a}\} \otimes (\{b\} \otimes \{\{c\}\} \cup \{\overline{b}\} \otimes \{\{d\}\})$$

Shannon decomposition: $\otimes$ operator on family of sets

Given the family of set F, G and sets S, U:
We can define an operator $\otimes$, we need to consider also $\overline{v}$ in $\cup$

$$S \otimes (F \cup G) = (S \otimes F) \cup (S \otimes G)$$

$$(S \cup \overline{S}) \otimes \{U\} = \{S \cup U\} \text{ if } \overline{S} = \emptyset \wedge S \neq \emptyset$$

$$(S \cup \overline{S}) \otimes \{U\} = \{U\} \text{ if } \overline{S} \neq \emptyset \wedge S = \emptyset$$

$$(S \cup \overline{S}) \otimes \{U\} = \{S \cup U, U\} \text{ if } \overline{S} \neq \emptyset \wedge S \neq \emptyset$$

$$(S \cup \overline{S}) \otimes \{U\} = \emptyset \text{ if } \overline{S} = \emptyset \wedge S = \emptyset$$

Examples:

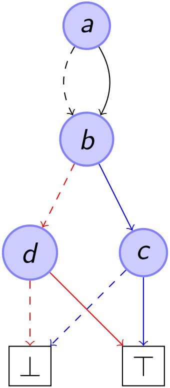
$$\{a, b\} \otimes \{\{c\}, \{d\}, \{d, e\}\} = \{\{a, b, c\}, \{a, b, d\}, \{a, b, d, e\}\}$$

$$\{a, b\} \otimes \{\{a, b, c\}, \{a, d\}, \{b, d\}, \{d\}\} = \{\{a, b, c\}, \{a, b, d\}\}$$

From Shannon decomposition to SFDD

$$\{a, \overline{a}\} \otimes (\{b\} \otimes \{\{c\}\} \cup \{\overline{b}\} \otimes \{\{d\}\})$$

is then transcoded into :



A set S over terms $T = \{t_1, t_2, ..., t_m\}$ is represented as a function f from T to $\mathbb{B}$, such as:

$$\begin{aligned} \forall s \in S, \quad & f_S(s) = t \\ \forall s \in T - S, \quad & f_S(s) = f \end{aligned}$$

A familly of sets $F = \{S_1, S_2, ..., S_n\}$, is defined as the following boolean function of arity m: $F : \mathbb{B} \times \mathbb{B} \times ... \times \mathbb{B} \to \mathbb{B}$ such as

$$\begin{aligned} \forall i \in 1...n, F(f_{S_i}(t_1), f_{S_i}(t_2), ..., f_{S_i}(t_m)) \quad & = t \\ otherwise \quad & = f \end{aligned}$$

Why Set Family Decision Diagrams?

Operations

This shows the correspondance between SFDD and BDD if we provide a total order over elements of T.

Although they are structurally similar, they benefit from different operations.

Moreover SFDD can be extended to other decision diagrams such as MFDD (encoding set of <KEY,VALUE>) and ΣDD (encoding set of Σ Terms) seamlessly.

Definition (Formal definition)

Let T be a set of terms. The set of SFDDs $\mathbb{S}$ is inductively defined by:

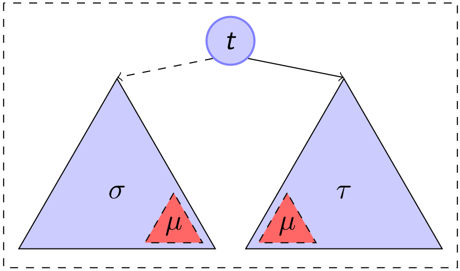
- $\bot \in \mathbb{S}$ is the rejecting terminal
- $\top \in \mathbb{S}$ is the accepting terminal
- $\langle t, \tau, \sigma \rangle \in \mathbb{S}$ if and only if $t \in T$, $\tau \in \mathbb{S}$, $\sigma \in \mathbb{S}$

sous arbre rejetant

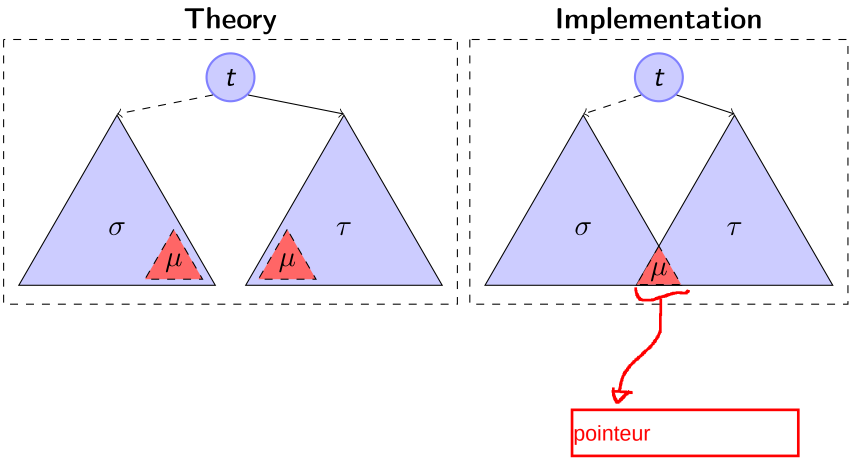
sous arbre acceptant

Theory VS Implementation

Theory

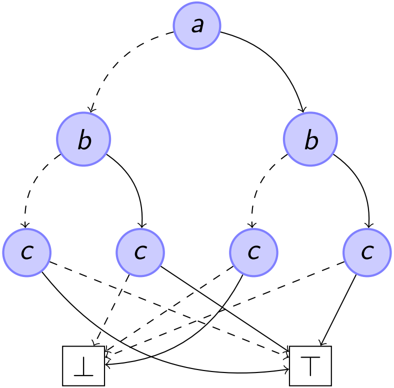


Theory VS Implementation



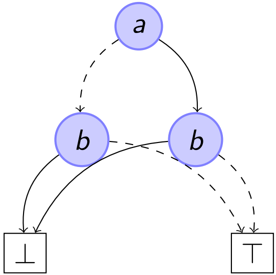
Set Family Decision Diagrams

Brute form



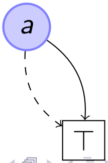
$S = \{\emptyset, \{c\}, \{c, b\}, \{c, b, a\}\}$

It is not optimal (neither unique, in fact depends on the constraint) as there is no common part (except the terminals) and several representation for the same set.



$$S = \{\emptyset, \{a\}\}$$

Representation uniqueness ?



From the brute ordered shape, we can reduce slightly the unnecessary nodes:

- remove negative nodes, i.e nodes with accept branch pointing to $\perp$, they are not providing any information.
- share common sub trees (not expressed in this formal definition)

clean : $\mathbb{S} \to \mathbb{S}$ removes a negative node from all sets that contain it:

$$\mathit{clean}(\perp) = \perp$$
$$\mathit{clean}(\top) = \top$$
$$\mathit{clean}(\langle t, \tau, \sigma \rangle) = \begin{cases} \mathit{clean}(\sigma) & \text{if } \tau = \perp \\ \langle t, \mathit{clean}(\tau), \mathit{clean}(\sigma) \rangle & \text{if } \mathit{otherwise} \end{cases}$$

NB: *clean* is an homomorphism.

Let $S \in \mathbb{S}$ be the SFDD $\langle t, \tau, \sigma \rangle$, we call τ its take node and σ its skip node.

S is canonical if for all its nodes, the skip node and take node represent greater terms or terminals, and no take node is the rejecting terminal. (sharing ?)

Definition (Canonical form)

Let T be a set of terms, and $< \in T \times T$ a total ordering on T. A SFDD $S \in \mathbb{S}$ is canonical if and only if

- S is the rejecting terminal $\bot$
- S is the accepting terminal $\top$
- $S = \langle t, \tau, \sigma \rangle$ where
  - $\tau = \langle t_\tau, \tau_\tau, \sigma_\tau \rangle \implies t < t_\tau$ and $\tau \neq \bot$
  - $\sigma = \langle t_\sigma, \tau_\sigma, \sigma_\sigma \rangle \implies t < t_\sigma$ ou $\sigma = \bot$ ou $\sigma = \top$
  - τ and σ are canonical

From the brute ordered shape, we can reduce by the *clean* operation. Shared trees are themselves describbed by the fact that equivalent subtrees are collapsed by an equivalence relation. $\equiv \subseteq \mathbb{S} \times \mathbb{S}$ identify similar sets:

$$\begin{aligned} \bot &\equiv \bot \\ \top &\equiv \top \\ \langle t, \tau, \sigma \rangle &\equiv \langle t, \tau', \sigma' \rangle && \text{if } \tau \equiv \tau' \wedge \sigma \equiv \sigma' \end{aligned}$$

The structure which is implemented is then $\mathbb{S} = \mathit{clean}(\mathbb{S}_{\mathit{brute}}) / \equiv$. Implementations share same subtrees and memorization can be used due to the functional nature of operations (no side effects).

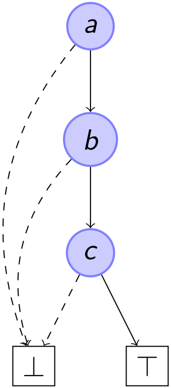
We give some basic examples of SFDD for a given set of sets from S and a total order $a < b < c$:

$$S = \{a, b, c\}$$

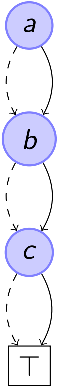
$$\wp(S) = \{\{a, b, c\}, \{a, b\}, \{b, c\}, \{a, c\}, \{a\}, \{b\}, \{c\}, \varnothing\}$$

$$\wp(S) - S = \{\{a, b\}, \{b, c\}, \{a, c\}, \{a\}, \{b\}, \{c\}, \varnothing\}$$

$$\wp(S) - \varnothing = \{\{a, b, c\}, \{a, b\}, \{b, c\}, \{a, c\}, \{a\}, \{b\}, \{c\}\}$$

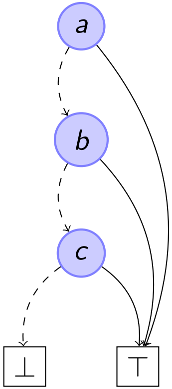


$enc(\{\{a, b, c\}\})$



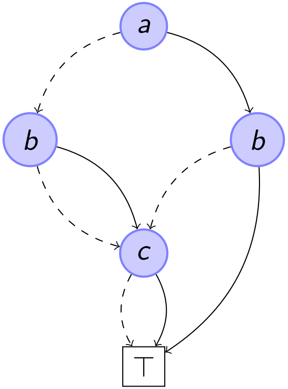
$enc(\wp(\{a, b, c\}))$

Set Family Decision Diagrams
Example



$$enc(\{\{a\},\{b\},\{c\}\})$$

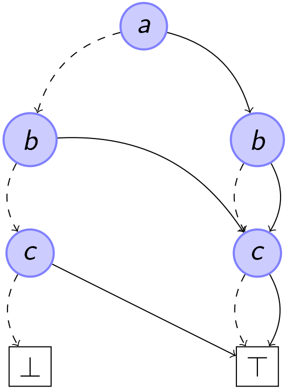
Set Family Decision Diagrams
Example



$enc(\wp(\{a, b, c\}) - \{a, b, c\})$
It is not a good case of encoding.

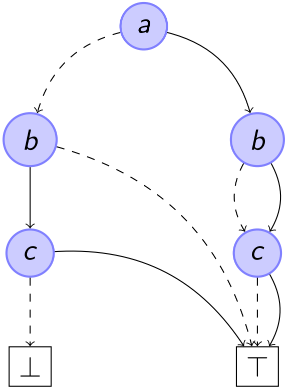
Set Family Decision Diagrams

Example



$enc(\wp(\{a, b, c\}) - \emptyset)$

It is one of the bad case we can expect, comparable to the previous one. But we can do worse.

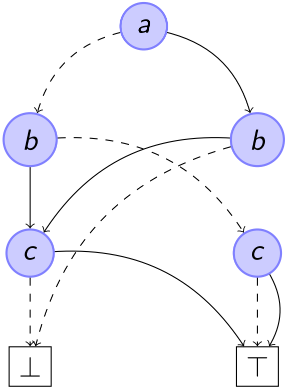


$enc(\{\{a, b, c\}, \{a, b\}, \{a, c\}, \{b, c\}, \{a\}, \emptyset\})$

It is one of the worst case we can expect if we remove also the non singleton sets, comparable to the previous one, we need $2^{|S|} - k$ nodes to encode $\wp(S) - \emptyset$.

Set Family Decision Diagrams

Example



$$S_{i+1} = \{\emptyset\} \cup (S_i \oplus \{e_{i+1}\}), 0 \leq i \leq n-1$$

$$S_0 = \{\emptyset\}$$

The union of two SFDDs is given by: $\{\{a,b\},\{c\}\} \cup \{\{c,d\},\{d\},\{\}\}$

$\mathrm{enc}(\{\}) = \bot$

$\mathrm{enc}(\{\emptyset\}) = \top$

$$A \cup B = B \cup A \quad = \{\{a,b\},\{c\},\{c,d\},\{d\}\}$$

$$A \cup A = A$$

$$\bot \cup A = A$$

$$\top \cup \langle t, \tau, \sigma \rangle = \langle t, \tau, \top \cup \sigma \rangle$$

$$\langle t, \tau, \sigma \rangle \cup \langle t', \tau', \sigma' \rangle = \begin{cases} \langle t, \tau, \sigma \cup \langle t', \tau', \sigma' \rangle \rangle & \text{if } t < t' \\ \langle t, \tau \cup \tau', \sigma \cup \sigma' \rangle & \text{if } t = t' \\ \langle t', \tau', \sigma' \cup \langle t, \tau, \sigma \rangle \rangle & \text{if } t > t' \end{cases}$$

The intersection of two SFDDs is given by:

avec les familles de s

$$A \cap B = B \cap A$$
$$A \cap A = A$$
$$\bot \cap A = \bot$$
$$\top \cap \langle t, \tau, \sigma \rangle = \top \cap \sigma$$
$$\langle t, \tau, \sigma \rangle \cap \langle t', \tau', \sigma' \rangle = \begin{cases} \sigma \cap \langle t', \tau', \sigma' \rangle & \text{if } t < t' \\ \langle t, \tau \cap \tau', \sigma \cap \sigma' \rangle & \text{if } t = t' \\ \langle t, \tau, \sigma \rangle \cap \sigma' & \text{if } t > t' \end{cases}$$

The encoding of a set into a SFDD is given by:

Famille

set

$$
\begin{aligned}
\mathrm{enc}(\varnothing) &= \bot \\
\mathrm{enc}(\{\varnothing\}) &= \top \\
\mathrm{enc}(S \cup \{s\}) &= \mathrm{enc}(S) \cup \mathrm{enc}(\{s\}) \\
t < \min(s) \implies \mathrm{enc}(\{s \cup \{t\}\}) &= \langle t, \mathrm{enc}(\{s\}), \bot \rangle
\end{aligned}
$$

=

encodage de {s}on commence par le min du set

Set Family Decision Diagrams

Decoding

The decoding of one SFDD is given by:

$\{\{b\},\{c,d\}\} \oplus a$

$= \{\{a,b\},\{a,c,d\}\}$

$$\begin{aligned}
\mathrm{dec}(\bot) &= \varnothing \\
\mathrm{dec}(\top) &= \{\varnothing\} \\
\mathrm{dec}(\langle t, \tau, \sigma \rangle) &= (\mathrm{dec}(\tau) \oplus t) \cup \mathrm{dec}(\sigma)
\end{aligned}$$

Where $\oplus$ is defined as follows:

$$\bigcup_{s \in S} \{s\} \oplus t = \bigcup_{s \in S} \{s \cup \{t\}\}$$

element de

famille

The decoding/encoding of one set is the identity (and the reverse):

$$\forall S \subseteq \mathcal{P}(T), \mathrm{dec}(\mathrm{enc}(S)) = S$$
$$\forall S \in \mathbb{S}, \mathrm{enc}(\mathrm{dec}(S)) = S$$

We write as index the reference set T for the encoding : enc_T

- Extending the reference set from T to T' ($T \subseteq T'$) does not imply changing the representation:
  $\forall S \subseteq \mathcal{P}(T) \Rightarrow \mathrm{enc}_T(S) = \mathrm{enc}_{T'}(S)$
- Under some constraint we can reduce the reference set $T' \subseteq T$, with or without change,$\forall S \subseteq \mathcal{P}(T)$:
  - case 1: $S \cap (T - T') = \emptyset \Rightarrow$
    $\mathrm{enc}_T(S) = \mathrm{enc}_{T'}(S)$
  - case 2: $S \cap (T - T') \neq \emptyset \Rightarrow$
    $\mathrm{enc}_T(S \cap T') = \mathrm{enc}_{T'}(S \cap T') = \mathrm{enc}_T(S) \ominus (T - T')$

Where $\cup, -$ and $\cap$ are defined as extension of set operation on family of sets, $\ominus$ is defined later on SFDD.

Homomorphisms are operations that preserve union:

$$\phi(S \cup S') = \phi(S) \cup \phi(S')$$

They also support operations that are themselves homomorphisms:

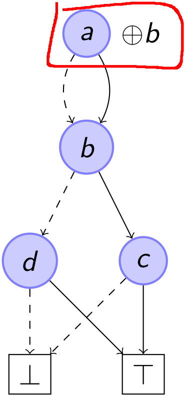
$$\begin{aligned}
\forall S, (\phi_1 + \phi_2)(S) &= \phi_1(S) \cup \phi_2(S) \\
\forall S, (\phi_1 \times \phi_2)(S) &= \phi_1(S) \cap \phi_2(S) \\
\forall S, (\phi_1 \circ \phi_2)(S) &= \phi_1(\phi_2(S))
\end{aligned}$$

$\oplus : \mathbb{S}, T \to \mathbb{S}$ inserts a term $a \in T$ into all sets of a SFDD:

$$\begin{aligned}
\bot \oplus a &= \bot \\
\top \oplus a &= \langle a, \top, \bot \rangle \\
\langle t, \tau, \sigma \rangle \oplus a &= \begin{cases} \langle t, \tau \oplus a, \sigma \oplus a \rangle & \text{if } t < a \\ \langle t, \tau \cup \sigma, \bot \rangle & \text{if } t = a \\ \langle a, \langle t, \tau, \sigma \rangle, \bot \rangle & \text{if } t > a \end{cases}
\end{aligned}$$

NB: $\oplus$ is an homomorphism.

Example: enc$(\{\{a, b, c\}, \{a, d\}, \{b, c\}, \{d\}\}) \oplus b$

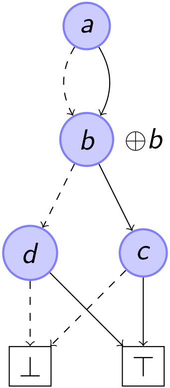


$b > a$ 1

$b = b$ 2

$b < d$, $b < c$ 3

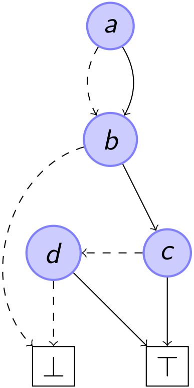
Example: $\mathrm{enc}(\{\{a,b,c\},\{a,d\},\{b,c\},\{d\}\}) \oplus b$



Set Family Decision Diagrams

Insertion

Example: $\mathrm{enc}(\{\{a,b,c\},\{a,d\},\{b,c\},\{d\}\}) \oplus b$



Encodes the sets:

- $\{a,b,c\}$ $\{b,c\}$ $\{b,d\}$
- $\{a,b,d\}$

Set Family Decision Diagrams

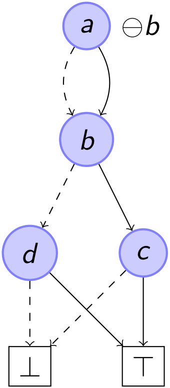
Removal

$\ominus : \mathbb{S}, T \to \mathbb{S}$ removes a term $a \in T$ from all sets that contain it:

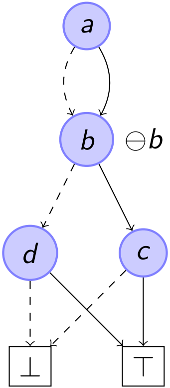
$$
\begin{aligned}
\bot \ominus a &= \bot \\
\top \ominus a &= \top \\
\langle t, \tau, \sigma \rangle \ominus a &= \begin{cases} \langle t, \tau \ominus a, \sigma \ominus a \rangle & \text{if } t < a \\ \sigma \cup \tau & \text{if } t = a \\ \langle t, \tau, \sigma \rangle & \text{if } t > a \end{cases}
\end{aligned}
$$

NB: $\ominus$ is an homomorphism.

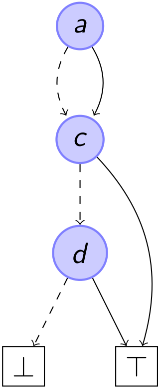
Example: $\mathrm{enc}(\{\{a,b,c\},\{a,d\},\{b,c\},\{d\}\}) \ominus b$



Example: $\mathrm{enc}(\{\{a, b, c\}, \{a, d\}, \{b, c\}, \{d\}\}) \ominus b$



Example: $\mathrm{enc}(\{\{a, b, c\}, \{a, d\}, \{b, c\}, \{d\}\}) \ominus b$



Encodes the sets:

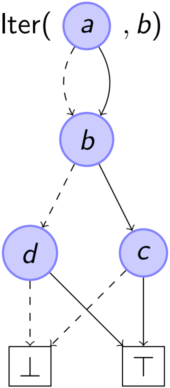
- $\{a, c\}$
- $\{a, d\}$
- $\{c\}$
- $\{d\}$

filter : $\mathbb{S}, T \to \mathbb{S}$ filters out the sets that don't contain a term $a \in T$:

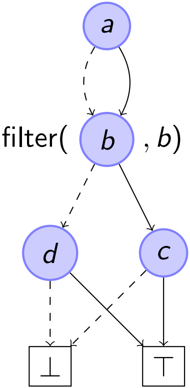
$$\mathrm{filter}(\bot, a) = \bot$$
$$\mathrm{filter}(\top, a) = \bot$$
$$\mathrm{filter}(\langle t, \tau, \sigma \rangle, a) = \begin{cases} \langle t, \mathrm{filter}(\tau, a), \mathrm{filter}(\sigma, a) \rangle & \text{if } t < a \\ \langle t, \tau, \bot \rangle & \text{if } t = a \\ \bot & \text{if } t > a \end{cases}$$

NB1: filter is an homomorphism.

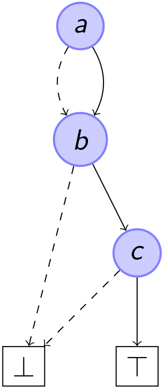
Example: filter(enc($\{\{a, b, c\}, \{a, d\}, \{b, c\}, \{d\}\}$), b)

filter( , b)

Example: filter(enc($\{\{a, b, c\}, \{a, d\}, \{b, c\}, \{d\}\}$), b)



Example: $\mathrm{filter}(\mathrm{enc}(\{\{a, b, c\}, \{a, d\}, \{b, c\}, \{d\}\}), b)$



Encodes the sets:

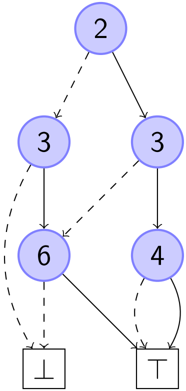
- $\{a, b, c\}$
- $\{b, c\}$

An inductive homomorphism is a tuple $\phi = \langle S, i \rangle$ where:

- $S \in \mathbb{S}$
- $i(A) = \langle \phi_\tau, \phi_\sigma \rangle$ where ϕ_τ, ϕ_σ are homomorphisms and $A \in \mathbb{S} \setminus \{\bot, \top\}$

Let $\phi = \langle S, i \rangle$, its application on $A \in \mathbb{S}$ is given by:

$$\phi(A) = \begin{cases} \bot & \text{if } A = \bot \\ S & \text{if } A = \top \\ \langle t, \phi_\tau(\tau), \phi_\sigma(\sigma) \rangle & \text{if } A = \langle t, \tau, \sigma \rangle, i(A) = \langle \phi_\tau, \phi_\sigma \rangle \end{cases}$$

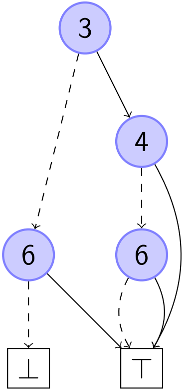


Example: removing values smaller than of 4

$$\phi = \langle \top, i \rangle$$

$$i(\langle t, \tau, \sigma \rangle) = \begin{cases} \langle h[\perp], \phi \circ (h[\tau] + \mathrm{id}) \rangle & \text{if } t < 4 \\ \langle \mathrm{id}, \mathrm{id} \rangle & \text{otherwise} \end{cases}$$

where $\forall S, \mathrm{id}(S) = S$ and $\forall S, h[K](S) = K$.

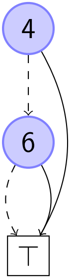


Example: removing values smaller than of 4

$$\phi = \langle \top, i \rangle$$

$$i(\langle t, \tau, \sigma \rangle) = \begin{cases} \langle h[\bot], \phi \circ (h[\tau] + \mathrm{id}) \rangle & \text{if } t < 4 \\ \langle \mathrm{id}, \mathrm{id} \rangle & \text{otherwise} \end{cases}$$

where $\forall S, \mathrm{id}(S) = S$ and $\forall S, h[K](S) = K$.



Example: removing values smaller than of 4

$$\phi = \langle \top, i \rangle$$

$$i(\langle t, \tau, \sigma \rangle) = \begin{cases} \langle h[\bot], \phi \circ (h[\tau] + \mathrm{id}) \rangle & \text{if } t < 4 \\ \langle \mathrm{id}, \mathrm{id} \rangle & \text{otherwise} \end{cases}$$

where $\forall S, \mathrm{id}(S) = S$ and $\forall S, h[K](S) = K$.

The count of members in a family is the operation size:

$$\mathrm{size}(S) = \begin{cases} 0 & \text{if } S = \bot \\ 1 & \text{if } S = \top \\ \mathrm{size}(\tau) + \mathrm{size}(\sigma) & \text{if } S = \langle t, \tau, \sigma \rangle \end{cases}$$

NB: size is not an homomorphism.

$$\begin{aligned} size(\{\{a,b\}\} \cup \{\{a,b\}\}) &= size(\{\{a,b\}\}) + size(\{\{a,b\}\}) \\ size(\{\{a,b\}\}) &= 2 + 2 \\ 2 &=? 4 \end{aligned}$$

- Encoding space for family of sets
- Operation on that space for usual manipulation of sets
- Efficient implementations based on sharing and memoization
- Homomorphisms for the respect of familly of sets
- Generalization to functions with finite domains

References I

[1] J. R. Burch, E. M. Clarke, K. L. McMillan, D. L. Dill, and L. J. Hwang. Symbolic model checking: 10^{20} states and beyond. *Inf. Comput.*, 98(2):142–170, 1992.

[2] Gianfranco Ciardo, Gerald Lüttgen, and Radu Siminiceanu. Efficient symbolic state-space construction for asynchronous systems. In *Application and Theory of Petri Nets 2000 (Proc. 21th Int. Conf. on Applications and Theory of Petri Nets, Aarhus, Denmark), Lecture Notes in Computer Science 1825*, pages 103–122. Springer-Verlag, 2000.

References II

[3] Jean-Michel Couvreur, Emmanuelle Encrenaz, Emmanuel Paviot-Adet, Denis Poitrenaud, and Pierre-André Wacrenier. Data decision diagrams for Petri net analysis. In *Proceedings of the 23th International Conference on Application and Theory of Petri Nets (ICATPN'02)*, volume 2360 of *Lecture Notes in Computer Science*, pages 101–120, Adelaide, Australia, June 2002. Springer Verlag.

[4] E. Allen Emerson and A. Prasad Sistla, editors. *Computer Aided Verification, 12th International Conference, CAV 2000, Chicago, IL, USA, July 15-19, 2000, Proceedings*, volume 1855 of *Lecture Notes in Computer Science*. Springer, 2000.

References III

[5] Patrice Godefroid. Using partial orders to improve automatic verification methods. In *CAV '90: Proceedings of the 2nd International Workshop on Computer Aided Verification*, pages 176–185, London, UK, 1991. Springer-Verlag.

[6] Patrice Godefroid and Pierre Wolper. A partial approach to model checking. In *Information and Computation*, pages 406–415, 1994.

[7] Robert P. Kurshan. *Computer-aided verification of coordinating processes: the automata-theoretic approach*. Princeton University Press, Princeton, NJ, USA, 1994.

References IV

[8] Alban Linard, Emmanuel Paviot-Adet, Fabrice Kordon, Didier Buchs, and Samuel Charron. polydd: Towards a framework generalizing decision diagrams. In *10th International Conference on Application of Concurrency to System Design, ACSD 2010, Braga, Portugal, 21-25 June 2010*, pages 124–133, 2010.

[9] Doron Peled. All from one, one for all: on model checking using representatives. In *CAV '93: Proceedings of the 5th International Conference on Computer Aided Verification*, pages 409–423, London, UK, 1993. Springer-Verlag.

[10] Antti Valmari. A stubborn attack on state explosion. *Form. Methods Syst. Des.*, 1(4):297–322, 1992.

References V

[11] Antti Valmari. The state explosion problem. In *Lectures on Petri Nets I: Basic Models, Advances in Petri Nets, the volumes are based on the Advanced Course on Petri Nets*, pages 429–528, London, UK, 1998. Springer-Verlag.