Reading group: formal methods for robust deep learning

Julien Girard

13 décembre 2018

Software safety

The goal of software safety is to ensure that we build the program good
With respect to a given **specification**

Software safety

The goal of software safety is to ensure that we build the program good

With respect to a given **specification**

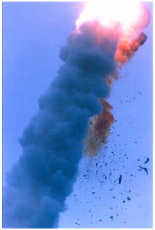


Figure – Preventing bugs on critical software

Figure – A mature field active both on academics and industry, and countless successes

Deep learning software specifics

- High number of variables and parameters (25 millions for ResNet-50, some have billions)

Deep learning software specifics

- High number of variables and parameters (25 millions for ResNet-50, some have billions)
- Execution environment is unknown (ex : computer vision)

Deep learning software specifics

- High number of variables and parameters (25 millions for ResNet-50, some have billions)
- Execution environment is unknown (ex : computer vision)
- Dangerous weaknesses, not clearly understood : adversarial examples, model theft, membership attacks

Deep learning software specifics

- High number of variables and parameters (25 millions for ResNet-50, some have billions)
- Execution environment is unknown (ex : computer vision)
- Dangerous weaknesses, not clearly understood : adversarial examples, model theft, membership attacks
- High-dimension geometric spaces are counterintuitive and makes analysis difficult

Deep learning software specifics

- High number of variables and parameters (25 millions for ResNet-50, some have billions)
- Execution environment is unknown (ex : computer vision)
- Dangerous weaknesses, not clearly understood : adversarial examples, model theft, membership attacks
- High-dimension geometric spaces are counterintuitive and makes analysis difficult

All of those makes it difficult for us to reuse bluntly our formal methods toolset

Abstract interpretation

Motivations

- Complete analysis can be expensive : `int i; char p[100]; ...; p[i]`

Motivations

- Complete analysis can be expensive : `int i; char p[100]; ...; p[i]`
- Sometimes, only partial knowledge is needed :
  `int i; ...; while (i>0); ...`

Intuition

- Key insight : *abstract* the program to produce a more easily computationable entity

Intuition

- Key insight : *abstract* the program to produce a more easily computationable entity
- Use this abstraction to exhibit interesting properties

Intuition

- Key insight : *abstract* the program to produce a more easily computationable entity
- Use this abstraction to exhibit interesting properties
- *This is an abstract interpretation*

Example : modulo function sign

```
int mod(int A, int B) {
    int Q = 0;
    int R = A;
    while (R >= B) {
        R = R - B;
        Q = Q + 1;
    }
    return R;
}
```

Example : modulo function sign

```
int mod(int A, int B) {
    int Q = 0;
    int R = A;
    while (R >= B) {
        R = R - B;
        Q = Q + 1;
    }
    return R;
}
```

Real semantic (a semantic is the set of all possible executions of a program) : $A = 10,\ B = 3$:

$$\langle l : a, b, q, r\rangle$$

$$\langle 1 : 10, 3\rangle \rightarrow \langle 2 : 10, 3, 0\rangle \rightarrow \langle 3 : 10, 3, 0, 10\rangle \rightarrow$$

$$\langle 4 : 10, 3, 0, 10\rangle \rightarrow \langle 5 : 10, 3, 0, 7\rangle \rightarrow \langle 6 : 10, 3, 1, 7\rangle \rightarrow$$

$$\langle 4 : 10, 3, 1, 7\rangle \rightarrow \langle 5 : 10, 3, 1, 4\rangle \rightarrow \langle 6 : 10, 3, 2, 4\rangle \rightarrow$$

$$\langle 4 : 10, 3, 2, 4\rangle \rightarrow \langle 5 : 10, 3, 2, 1\rangle \rightarrow \langle 6 : 10, 3, 3, 1\rangle \rightarrow \langle 7 : 10, 3, 3, 1\rangle \rightarrow$$

Example : modulo function sign

```
int mod(int A, int B) {
    int Q = 0;
    int R = A;
    while (R >= B) {
        R = R - B;
        Q = Q + 1;
    }
    return R;
}
```

Abstract semantic : $A \geq 0$, $B \geq 0$:

$$\langle l : a, b, q, r \rangle$$

$$\langle 1 : (\geq 0), (\geq 0) \rangle \to \langle 2 : (\geq 0), (\geq 0), 0 \rangle \to \langle 3 : (\geq 0), (\geq 0), 0, (\geq 0) \rangle \to$$
$$\langle 4 : (\geq 0), (\geq 0), 0, (\geq 0) \rangle \to \langle 5 : (\geq 0), (\geq 0), 0, \top \rangle \to \langle 6 : (\geq 0), (\geq 0), (\geq 0), \top \rangle \to$$
$$\langle 4 : (\geq 0), (\geq 0), (\geq 0), \top \rangle \to \langle 5 : (\geq 0), (\geq 0), (\geq 0), \top \rangle \to \langle 6 : (\geq 0), (\geq 0), (\geq 0), \top \rangle \to$$
$$\langle 7 : (\geq 0), (\geq 0), (\geq 0), \top \rangle$$

Example : modulo function sign

```
int mod(int A, int B) {
    int Q = 0;
    int R = A;
    while (R >= B) {
        R = R - B;
        Q = Q + 1;
    }
    return R;
}
```

$\top$ R **because** $R - B =^{\#} (\geq 0) - (\geq 0) = \top$

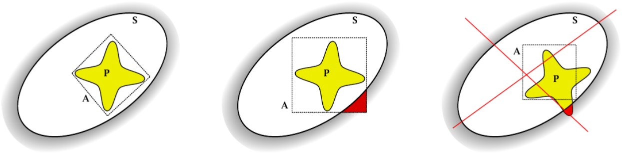
$R \geq B =^{\#} \top \geq (\geq 0)$

Example : modulo function sign

```
int mod(int A, int B) {
    int Q = 0;
    int R = A;
    while (R >= B) {
        R = R - B;
        Q = Q + 1;
    }
    return R;
}
```

Loop invariant : $\langle(\geq 0), (\geq 0), (\geq 0), \top\rangle$

What is at stake ?



precise analysis $A \subseteq S \implies P \subseteq S$ (a)

false alarm $A \not\subseteq S$ but $P \subseteq S$ (b)

unsound analysis $A \subseteq S$ but $P \not\subseteq S$ (c)

Figure 1.6: Proving that a program P satisfies a safety specification S, i.e., that $P \subseteq S$, using an abstraction A of P: (a) succeeds, (b) fails with a false alarm, and (c) is not a possible configuration for a sound analysis.

Figure – Figure comes from Antoine Minet tutorial

Balance between relevant properties, computationable executions and accuracy of abstraction

Partial order relations

A partial order $\sqsubseteq$ on a set X is a relation holding :

1. reflexivity : $\forall x \in X : x \sqsubseteq x$;
2. anti-symetric : $\forall x, y \in X : (x \sqsubseteq y) \wedge (y \sqsubseteq x) \Rightarrow x = y$;
3. transitivity : $\forall x, y, z \in X : (x \sqsubseteq y) \wedge (y \sqsubseteq z) \Rightarrow x \sqsubseteq z$;

Partial, means sometimes there are some x and y not sharing any order relation.

Partial order relations

A partial order $\sqsubseteq$ on a set X is a relation holding :

1. reflexivity : $\forall x \in X : x \sqsubseteq x$;
2. anti-symetric : $\forall x, y \in X : (x \sqsubseteq y) \wedge (y \sqsubseteq x) \Rightarrow x = y$;
3. transitivity : $\forall x, y, z \in X : (x \sqsubseteq y) \wedge (y \sqsubseteq z) \Rightarrow x \sqsubseteq z$;

Partial, means sometimes there are some x and y not sharing any order relation.
$\sqcup$ and $\sqcap$ are resp. *lowerupperbound* and *greaterlowerbound* of two elements of any subset of X. If they exist, they are unique

Partial order relations

A partial order $\sqsubseteq$ on a set X is a relation holding :

1. reflexivity : $\forall x \in X : x \sqsubseteq x$;
2. anti-symetric : $\forall x, y \in X : (x \sqsubseteq y) \wedge (y \sqsubseteq x) \Rightarrow x = y$;
3. transitivity : $\forall x, y, z \in X : (x \sqsubseteq y) \wedge (y \sqsubseteq z) \Rightarrow x \sqsubseteq z$;

Partial, means sometimes there are some x and y not sharing any order relation.
$\sqcup$ and $\sqcap$ are resp. *lowerupperbound* and *greaterlowerbound* of two elements of any subset of X. If they exist, they are unique
Most relevant partial order : **partial inclusion** $\subseteq$

Partial order relations

A partial order $\sqsubseteq$ on a set X is a relation holding :

1. reflexivity : $\forall x \in X : x \sqsubseteq x$;
2. anti-symetric : $\forall x, y \in X : (x \sqsubseteq y) \wedge (y \sqsubseteq x) \Rightarrow x = y$;
3. transitivity : $\forall x, y, z \in X : (x \sqsubseteq y) \wedge (y \sqsubseteq z) \Rightarrow x \sqsubseteq z$;

Partial, means sometimes there are some x and y not sharing any order relation.
$\sqcup$ and $\sqcap$ are resp. *lowerupperbound* and *greaterlowerbound* of two elements of any subset of X. If they exist, they are unique
Most relevant partial order : **partial inclusion** $\subseteq$

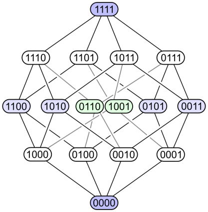


Figure – Partial order representation : Hasse Diagram

Lattice

A lattice is a partially ordered set X such as :

1. $\forall A \subseteq X : \sqcup A$ exist
2. $\forall A \subseteq X : \sqcap A$ exist
3. X as a smaller element $\bot$
4. X as a greatest element $\top$

Fixpoints

Définition

*A **fixpoint** for a function f is a point* x_{fixe} *such as* $f(x_{fixe}) = x_{fixe}$

Note that $f(x) \subseteq x$. A fixpoint is an *execution invariant*.

Fixpoints

Définition

*A **fixpoint** for a function f is a point x_{fixe} such as $f(x_{fixe}) = x_{fixe}$*

Note that $f(x) \subseteq x$. A fixpoint is an *execution invariant*.

Théorème (Knaster-Tarski fixpoint theorem)

If X is a complete lattice and $f : X \rightarrow X$ a monotonous application, then the ordered subset of all fixpoints of f is a non-empty complete lattice. In particular, f has a smaller and greater fixpoint.

Partial order and analysis

1. *approximation* with sound but non-comparable analysis
2. *valid regarding a specification* : a program semantic P respect a given specification S if $P \subseteq S$
3. *Sound analysis* : abstract semantic is coarser than real semantic
4. *Convergence* : order is necessary to have convergence towards a fixpoint

Let's summarize

- Lattice X : set with partial order relation $\subseteq$, a smallest element $\perp$ and a biggest element $\top$
- A fonction f is monotonous on X $\Rightarrow$, fixpoints x_{fixe} exists

Let's summarize

- Lattice X : set with partial order relation $\subseteq$, a smallest element $\perp$ and a biggest element $\top$
- A fonction f is monotonous on $X \Rightarrow$, fixpoints x_{fixe} exists

$X \rightarrow$?
$f \rightarrow$?
$x_{fixe} \rightarrow$?

Let's summarize

- Lattice X : set with partial order relation $\subseteq$, a smallest element $\perp$ and a biggest element $\top$
- A fonction f is monotonous on $X \Rightarrow$, fixpoints x_{fixe} exists

$X \rightarrow$ **The abstract semantic of a program**
$f \rightarrow$ **An evaluation on the abstract semantic**
$x_{fixe} \rightarrow$ **A snapshot of all the states of a program in the abstract semantic**

What is the frame of our lattice ?

A program state after an abstract execution

What is the frame of our lattice ?

A program state after an abstract execution

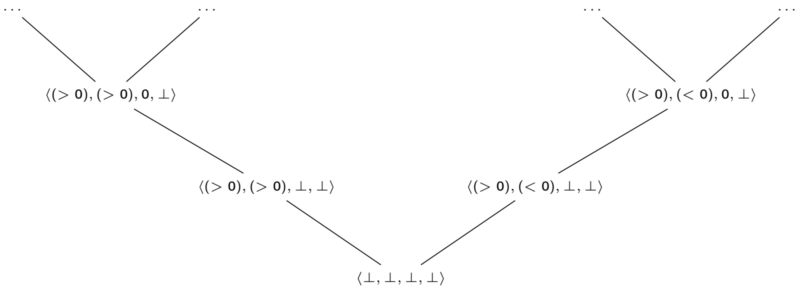


Figure – Partial Hasse diagram of the modulo fonction, for the abstract semantic of signs

Consequences

If we have a monotonous f (abstract evaluations), fixpoints (knowing program states in the abstract semantic) exists ! And we can compute them

Consequences

If we have a monotonous f (abstract evaluations), fixpoints (knowing program states in the abstract semantic) exists ! And we can compute them
Goal now : “monotonous” computations.

Definitions

Définition

Let D a domain.

- *an abstraction function $\alpha : P(\mathcal{R}^d \to D)$*
- *a concretization function $\gamma : D \to P(\mathcal{R}^d)$*

$d \in D$ is an abstraction of $P(\mathcal{R}^d)$, and $\gamma(d)$ gives us the corresponding values in $P(\mathcal{R}^d)$.

Théorème (Validity of abstract interpretation)

An abstract domain D is "sound" iff $X \subseteq \gamma(\alpha(X)) \forall X \subseteq \mathcal{R}^d$

Transfer functions

Let a function $f : \mathcal{R}^p \rightarrow \mathcal{R}^{d'}$. An abstract transformer is a function $T_f^{\#} : D \rightarrow D^{'}$ such as $f(\gamma(d)) \subseteq \gamma^{'}(T_f^{\#}(d))$ for all $d \in D$.

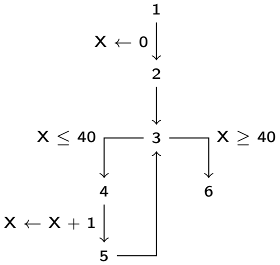
An abstract domain : intervals

Let $x \in \mathcal{R}^d, \varepsilon \in \mathcal{R}^d$. $[x - \varepsilon, x + \varepsilon]$ is an interval, also noted $[a, b]$. Transfer functions :

$$
\begin{aligned}
[a, b] + [c, d] = & \quad [a + b, c + d] \\
[a, b] * [c, d] = & \quad [a * b, c * d] \\
[a, b] = & \quad [-b, -a]
\end{aligned}
$$

An example of intervals

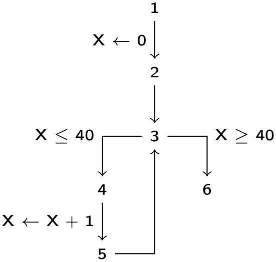
```
X <- 0
 while (X<40)
    X <- X+1
```



I	$\chi_I^{\#0}$	$\chi_I^{\#4}$	$\chi_I^{\#5}$	$\chi_I^{\#7}$	$\chi_I^{\#8}$

An example of intervals

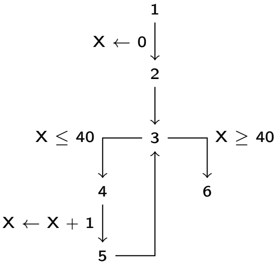
```
X <- 0
while (X<40)
    X <- X+1
```



l	$\chi_l^{\#0}$	$\chi_l^{\#4}$	$\chi_l^{\#5}$	$\chi_l^{\#7}$	$\chi_l^{\#8}$
1	$\top$	$\top$	$\top$	$\top$	$\top$

An example of intervals

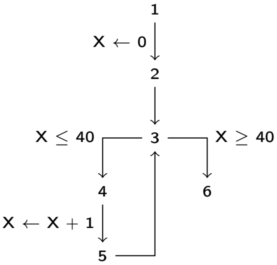
```
X <- 0
while (X<40)
    X <- X+1
```



l	$\chi_l^{\#0}$	$\chi_l^{\#4}$	$\chi_l^{\#5}$	$\chi_l^{\#7}$	$\chi_l^{\#8}$
1	$\top$	$\top$	$\top$	$\top$	$\top$
2	$\bot$	0	0	0	0

An example of intervals

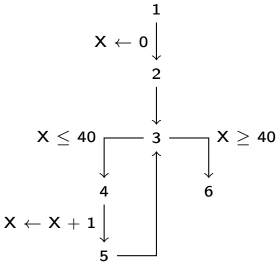
```
X <- 0
while (X<40)
    X <- X+1
```



I	$\chi_I^{\#0}$	$\chi_I^{\#4}$	$\chi_I^{\#5}$	$\chi_I^{\#7}$	$\chi_I^{\#8}$
1	$\top$	$\top$	$\top$	$\top$	$\top$
2	$\perp$	0	0	0	0
3	$\perp$	0	$[0, +\infty]$	$[0, +\infty]$	$[0, +\infty]$

An example of intervals

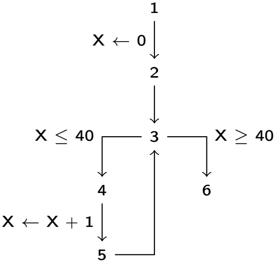
```
X <- 0
while (X<40)
    X <- X+1
```



l	$\chi_l^{\#0}$	$\chi_l^{\#4}$	$\chi_l^{\#5}$	$\chi_l^{\#7}$	$\chi_l^{\#8}$
1	$\top$	$\top$	$\top$	$\top$	$\top$
2	$\perp$	0	0	0	0
3	$\perp$	0	$[0, +\infty]$	$[0, +\infty]$	$[0, +\infty]$
4	$\perp$	0	0	$[0, 39]$	$[0, 39]$

An example of intervals

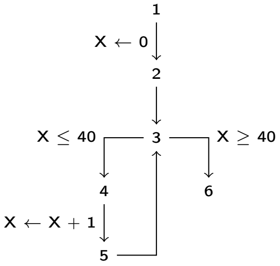
```
X <- 0
while (X<40)
    X <- X+1
```



l	$\chi_l^{\#0}$	$\chi_l^{\#4}$	$\chi_l^{\#5}$	$\chi_l^{\#7}$	$\chi_l^{\#8}$
1	$\top$	$\top$	$\top$	$\top$	$\top$
2	$\bot$	0	0	0	0
3	$\bot$	0	$[0, +\infty]$	$[0, +\infty]$	$[0, +\infty]$
4	$\bot$	0	0	$[0, 39]$	$[0, 39]$
5	$\bot$	1	1	$[1, 40]$	$[1, 40]$

An example of intervals

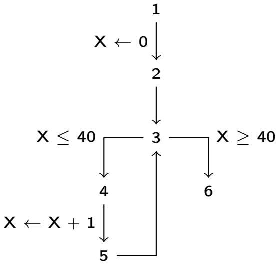
```
X <- 0
while (X<40)
    X <- X+1
```



I	$\chi_I^{\#0}$	$\chi_I^{\#4}$	$\chi_I^{\#5}$	$\chi_I^{\#7}$	$\chi_I^{\#8}$
1	$\top$	$\top$	$\top$	$\top$	$\top$
2	$\perp$	0	0	0	0
3	$\perp$	0	$[0, +\infty]$	$[0, +\infty]$	$[0, +\infty]$
4	$\perp$	0	0	$[0, 39]$	$[0, 39]$
5	$\perp$	1	1	$[1, 40]$	$[1, 40]$
6	$\perp$	$\perp$	$\perp$	$[40, +\infty]$	$[40, \infty]$

An example of intervals

```
X <- 0
while (X<40)
    X <- X+1
```



l	$\chi_l^{\#0}$	$\chi_l^{\#4}$	$\chi_l^{\#5}$	$\chi_l^{\#7}$	$\chi_l^{\#8}$
1	$\top$	$\top$	$\top$	$\top$	$\top$
2	$\bot$	0	0	0	0
3	$\bot$	0	$[0, +\infty]$	$[0, +\infty]$	$[0, +\infty]$
4	$\bot$	0	0	$[0, 39]$	$[0, 39]$
5	$\bot$	1	1	$[1, 40]$	$[1, 40]$
6	$\bot$	$\bot$	$\bot$	$[40, +\infty]$	$[40, \infty]$

Limitations : $x := [-1, 1], x - x = [-2, 2]$

Summary of work

1. build an abstraction of neural network using the abstract interpretation framework
2. encapsulate adversarial perturbations inside abstract domains
3. build robustness properties on abstract domains and learn networks to minimize adversarial loss

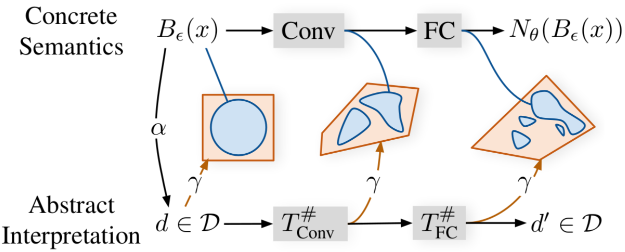


Figure – DiffAI/DeepZ control flow

Abstract domains used

- Intervals $[x-\varepsilon, x+\varepsilon]$
- Zonotopes (polytope with a symetry center) $z=(z_C, z_E)$, $z_C \in \mathcal{R}^d$ center, $z_E \in \mathcal{R}^{d*m}$ linear constraints
- Hybrids zonotope $h=\langle h_C, h_B, h_E \rangle$, $h_C \in \mathcal{R}^d$ center, $h_B \in \mathcal{R}^d_{\geq 0}$ perturbations, $h_E \in \mathcal{R}^{d*l}$ errors coefficients

Abstractions and concretizations

$$\gamma_H(h) = \{h_{conc}(\beta, e) | \beta \in [-1, 1]^d, e \in [-1, 1]^{d*m}\},$$
$$h_{conc} = h_C + diag(h_B) * \beta + h_E * e$$

Abstractions and concretizations

$\gamma_H(h) = \{h_{conc}(\beta, e) | \beta \in [-1,1]^d, e \in [-1,1]^{d*m}\}$,
$h_{conc} = h_C + diag(h_B) * \beta + h_E * e$
i-th total error of an hybrid zonotope h : $\varepsilon_H(h)_i = (h_B)_i + \sum_{j=1}^{m} |(h_E)_{i,j}|$
Interval concretization : $\iota_H(h)_i \, [(h_C)_i - \varepsilon_H(h)_i, (h_C)_i + \varepsilon_H(h)_i]$

Matrix operations

For a matrix M : $T_f^{\#}(h) = \langle M \cdot h_C, M \cdot h_B, M \cdot h_E \rangle$
Includes sum, scalar multiplication, convolutions. . .

ReLu

Let a zonotope z. A zonotope $z' = T^{\#(transfo)}_{Relu_i}$ with $m' = m + 1$ is computed : **zBox** If $min(\iota(z)) \geq 0$, ReLu has no effect and propagated zonotope is the same (modulo dimension). Else :

$(z'_C)_t = (z_C)_t$ for $t \neq i$
$(z'_E)_t = (z_E)_t$ for $t \neq i$
$(z'_C)_i = ReLu(\frac{1}{2} max(\iota(z)_i))$
$(z'_E)_{i,l} = 0$ for $l \leq m$
$(z'_E)_{i,m+1} = ReLu(\frac{1}{2} max(\iota(z)_i))$
$(z'_E)_{j,m+1} = 0$ for $j \leq i$

zDiag If $min(\iota(z)_i) \leq 0 \leq max(\iota(z)_i)$ holds, then : $(z'_C)_t = (z_C)_t$ for $t \neq i$
$(z'_E)_t = (z_E)_t$ for $t \neq i$
$(z'_C)_i = (z_C)_i z'_E)_{i,l}$
$(z'_E)_{i,l} = z_E)_{i,l}$ for $l \leq m$
$(z'_E)_{i,m+1} = -\frac{1}{2} min(\iota(z)_i)$
$(z'_E)_{j,m+1} = 0$ for $j \leq i$

Else, zBox

Adversarial training

Loss : $L(z,y) = max_{y' \neq y}(z_{y'} - z_y$, where z points and y labels.
Then the adversarial loss when minimized shows the π-robustness of all the training set :

$$L_N^A(x,y) = \max_{\tilde{z} \in \gamma(T_N^{\#}(\alpha(\pi(x))))} L(\tilde{z}, y).$$

Results

- Epoch training time multiplied between 3 and 7. An epoch on a baseline Resnet is 3.7s, against 12.6s with their method
- Test against one attack (PGD, Madry et al.)
- MNIST : 5.8% on adversarial test error, baseline 100%
- CIFAR-10 : ResNet with adversarial training has a 47.8% test error, baseline is 88%.

Conclusion

An elegant method combining the best of the two worlds, promising results but need to be compared against more attacks and with different metrics

Questions ?

:)

SMT

Boolean calculus

Two possible values : *false*(0) and *true*(1)

Boolean calculus

Two possible values : *false*(0) and *true*(1)
Rules are "good" :

- associativity : $A \wedge (B \wedge C) = (A \wedge B) \wedge C$
- commutativity $(A \wedge B = B \wedge A)$
- idempotency $(A \wedge A = A)$
- neutral elements : 1 for $\wedge$, 0 for $\vee$
- absorbant elements : 0 for $\wedge$, 1 for $\vee$
- distributivity

Boolean calculus

Two possible values : *false*(0) and *true*(1)
Rules are "good" :

- associativity : $A \wedge (B \wedge C) = (A \wedge B) \wedge C$
- commutativity ($A \wedge B = B \wedge A$)
- idempotency ($A \wedge A = A$)
- neutral elements : 1 for $\wedge$, 0 for $\vee$
- absorbant elements : 0 for $\wedge$, 1 for $\vee$
- distributivity

Some axioms

1. negation $\neg$
2. Morgan's law : $\neg(A \wedge B) = \neg A \vee \neg B$, same idea for $\vee$

Boolean calculus (following)

Vocabulaire :

- Litterals : elementary signs (values, variables)
- Clause (or term) : litterals disjunction (a$\vee$ b)
- A unit clause iff there is only one litteral involved
- Cunjonctive Normal Form : ((a$\vee$ b)$\wedge$ (b $\vee$ d))

Boolean calculus (following)

Vocabulaire :

- Litterals : elementary signs (values, variables)
- Clause (or term) : litterals disjunction (a$\vee$ b)
- A unit clause iff there is only one litteral involved
- Cunjonctive Normal Form : ((a$\vee$ b)$\wedge$ (b $\vee$ d))

Boolean calculus is used to encode logic formulae

SAT problem

- **Let a formula $A(x_1, x_2, \ldots, x_n)$, are there boolean values x_i making A true ?** : SAT
- **Let a formula $A(x_1, x_2, \ldots, x_n)$, is A true for all x_i ?** : VALID

VALID(A) is equivalent to $\neg$SAT($\neg A$)

SAT problem

- **Let a formula** $A(x_1, x_2, \ldots, x_n)$**, are there boolean values** x_i **making** A **true ?** : SAT
- **Let a formula** $A(x_1, x_2, \ldots, x_n)$**, is** A **true for all** x_i **?** : VALID

VALID(A) is equivalent to $\neg$SAT($\neg A$) NP-complete problem

Conflict Driven Clause Learning

Principle :

1. Look for a term leading the formula to UNSAT by assigning values iteratively to variables
2. Identify the origin of conflict and learn a clause preventing it
3. Repeat until SAT, TIMEOUT or UNSAT

Illustration

$$\varphi_1 = x_1 \vee x_4$$
$$\varphi_2 = x_1 \vee \overline{x_3} \vee \overline{x_8}$$
$$\varphi_3 = x_1 \vee x_8 \vee x_{12}$$
$$\varphi_4 = x_2 \vee x_{11}$$
$$\varphi_5 = \overline{x_3} \vee \overline{x_7} \vee x_{13}$$
$$\varphi_6 = \overline{x_3} \vee \overline{x_7} \vee \overline{x_{13}} \vee x_9$$
$$\varphi_7 = x_8 \vee \overline{x_7} \vee \overline{x_9}$$

Illustration

$$\varphi_1 = x_1 \vee x_4$$
$$\varphi_2 = x_1 \vee \overline{x_3} \vee \overline{x_8}$$
$$\varphi_3 = x_1 \vee x_8 \vee x_{12}$$
$$\varphi_4 = x_2 \vee x_{11}$$
$$\varphi_5 = \overline{x_3} \vee \overline{x_7} \vee x_{13}$$
$$\varphi_6 = \overline{x_3} \vee \overline{x_7} \vee \overline{x_{13}} \vee x_9$$
$$\varphi_7 = x_8 \vee \overline{x_7} \vee \overline{x_9}$$

$$x_1 \Rightarrow x_4 \; [\varphi_1]$$

Illustration

$$\varphi_1 = x_1 \vee x_4$$
$$\varphi_2 = x_1 \vee \overline{x_3} \vee \overline{x_8}$$
$$\varphi_3 = x_1 \vee x_8 \vee x_{12}$$
$$\varphi_4 = x_2 \vee x_{11}$$
$$\varphi_5 = \overline{x_3} \vee \overline{x_7} \vee x_{13}$$
$$\varphi_6 = \overline{x_3} \vee \overline{x_7} \vee \overline{x_{13}} \vee x_9$$
$$\varphi_7 = x_8 \vee \overline{x_7} \vee \overline{x_9}$$

$$x_1 \Rightarrow x_4 \; [\varphi_1]$$

Illustration

$$\varphi_1 = x_1 \vee x_4$$
$$\varphi_2 = x_1 \vee \overline{x_3} \vee \overline{x_8}$$
$$\varphi_3 = x_1 \vee x_8 \vee x_{12}$$
$$\varphi_4 = x_2 \vee x_{11}$$
$$\varphi_5 = \overline{x_3} \vee \overline{x_7} \vee x_{13}$$
$$\varphi_6 = \overline{x_3} \vee \overline{x_7} \vee \overline{x_{13}} \vee x_9$$
$$\varphi_7 = x_8 \vee \overline{x_7} \vee \overline{x_9}$$

$$x_1 \Rightarrow x_4 \ [\varphi_1]$$
$$x_3 \Rightarrow x_8 \ [\varphi_2],\ x_{12} \ [\varphi_3]$$

Illustration

$$\varphi_1 = x_1 \vee x_4$$
$$\varphi_2 = x_1 \vee \overline{x_3} \vee \overline{x_8}$$
$$\varphi_3 = x_1 \vee x_8 \vee x_{12}$$
$$\varphi_4 = x_2 \vee x_{11}$$
$$\varphi_5 = \overline{x_3} \vee \overline{x_7} \vee x_{13}$$
$$\varphi_6 = \overline{x_3} \vee \overline{x_7} \vee \overline{x_{13}} \vee x_9$$
$$\varphi_7 = x_8 \vee \overline{x_7} \vee \overline{x_9}$$

$$x_1 \Rightarrow x_4 \ [\varphi_1]$$
$$x_3 \Rightarrow x_8 \ [\varphi_2], \ x_{12} \ [\varphi_3]$$

Illustration

$$\begin{aligned}
\varphi_1 &= x_1 \vee x_4\\
\varphi_2 &= x_1 \vee \overline{x_3} \vee \overline{x_8}\\
\varphi_3 &= x_1 \vee x_8 \vee x_{12}\\
\varphi_4 &= x_2 \vee x_{11}\\
\varphi_5 &= \overline{x_3} \vee \overline{x_7} \vee x_{13}\\
\varphi_6 &= \overline{x_3} \vee \overline{x_7} \vee \overline{x_{13}} \vee x_9\\
\varphi_7 &= x_8 \vee \overline{x_7} \vee \overline{x_9}
\end{aligned}$$

$$x_1 \Rightarrow x_4\ [\varphi_1]$$
$$x_3 \Rightarrow x_8\ [\varphi_2],\ x_{12}\ [\varphi_3]$$

Illustration

$$\varphi_1 = x_1 \vee x_4$$
$$\varphi_2 = x_1 \vee \overline{x_3} \vee \overline{x_8}$$
$$\varphi_3 = x_1 \vee x_8 \vee x_{12}$$
$$\varphi_4 = x_2 \vee x_{11}$$
$$\varphi_5 = \overline{x_3} \vee \overline{x_7} \vee x_{13}$$
$$\varphi_6 = \overline{x_3} \vee \overline{x_7} \vee \overline{x_{13}} \vee x_9$$
$$\varphi_7 = x_8 \vee \overline{x_7} \vee \overline{x_9}$$

$$x_1 \Rightarrow x_4 \; [\varphi_1]$$
$$x_3 \Rightarrow x_8 \; [\varphi_2], \; x_{12} \; [\varphi_3]$$

Illustration

$$\varphi_1 = x_1 \vee x_4$$
$$\varphi_2 = x_1 \vee \overline{x_3} \vee \overline{x_8}$$
$$\varphi_3 = x_1 \vee x_8 \vee x_{12}$$
$$\varphi_4 = x_2 \vee x_{11}$$
$$\varphi_5 = \overline{x_3} \vee \overline{x_7} \vee x_{13}$$
$$\varphi_6 = \overline{x_3} \vee \overline{x_7} \vee \overline{x_{13}} \vee x_9$$
$$\varphi_7 = x_8 \vee \overline{x_7} \vee \overline{x_9}$$

$$x_1 \Rightarrow x_4 \; [\varphi_1]$$
$$x_3 \Rightarrow x_8 \; [\varphi_2], \; x_{12} \; [\varphi_3]$$
$$x_2 \Rightarrow x_{11} \; [\varphi_4]$$

Illustration

$\varphi_1 = x_1 \vee x_4$

$\varphi_2 = x_1 \vee \overline{x_3} \vee \overline{x_8}$

$\varphi_3 = x_1 \vee x_8 \vee x_{12}$

$\varphi_4 = x_2 \vee x_{11}$

$\varphi_5 = \overline{x_3} \vee \overline{x_7} \vee x_{13}$

$\varphi_6 = \overline{x_3} \vee \overline{x_7} \vee \overline{x_{13}} \vee x_9$

$\varphi_7 = x_8 \vee \overline{x_7} \vee \overline{x_9}$

$x_1 \Rightarrow x_4 \; [\varphi_1]$

$x_3 \Rightarrow x_8 \; [\varphi_2],\; x_{12} \; [\varphi_3]$

$x_2 \Rightarrow x_{11} \; [\varphi_4]$

Illustration

$\varphi_1 = x_1 \vee x_4$

$\varphi_2 = x_1 \vee \overline{x_3} \vee \overline{x_8}$

$\varphi_3 = x_1 \vee x_8 \vee x_{12}$

$\varphi_4 = x_2 \vee x_{11}$

$\varphi_5 = \overline{x_3} \vee \overline{x_7} \vee x_{13}$

$\varphi_6 = \overline{x_3} \vee \overline{x_7} \vee \overline{x_{13}} \vee x_9$

$\varphi_7 = x_8 \vee \overline{x_7} \vee \overline{x_9}$

$x_1 \Rightarrow x_4\ [\varphi_1]$

$x_3 \Rightarrow x_8\ [\varphi_2],\ x_{12}\ [\varphi_3]$

$x_2 \Rightarrow x_{11}\ [\varphi_4]$

$x_7 \Rightarrow x_{13}\ [\varphi_5],\ x_9\ [\varphi_6],\ \overline{x_9}\ [\varphi_7]$

Illustration

$$\varphi_1 = x_1 \lor x_4$$
$$\varphi_2 = x_1 \lor \overline{x_3} \lor \overline{x_8}$$
$$\varphi_3 = x_1 \lor x_8 \lor x_{12}$$
$$\varphi_4 = x_2 \lor x_{11}$$
$$\varphi_5 = \overline{x_3} \lor \overline{x_7} \lor x_{13}$$
$$\varphi_6 = \overline{x_3} \lor \overline{x_7} \lor \overline{x_{13}} \lor x_9$$
$$\varphi_7 = x_8 \lor \overline{x_7} \lor \overline{x_9}$$

$$x_1 \Rightarrow x_4\ [\varphi_1]$$
$$x_3 \Rightarrow x_8\ [\varphi_2],\ x_{12}\ [\varphi_3]$$
$$x_2 \Rightarrow x_{11}\ [\varphi_4]$$
$$x_7 \Rightarrow x_{13}\ [\varphi_5],\ x_9\ [\varphi_6],\ \overline{x_9}\ [\varphi_7]$$

Illustration

$$\varphi_1 = x_1 \vee x_4$$
$$\varphi_2 = x_1 \vee \overline{x_3} \vee \overline{x_8}$$
$$\varphi_3 = x_1 \vee x_8 \vee x_{12}$$
$$\varphi_4 = x_2 \vee x_{11}$$
$$\varphi_5 = \overline{x_3} \vee \overline{x_7} \vee x_{13}$$
$$\varphi_6 = \overline{x_3} \vee \overline{x_7} \vee \overline{x_{13}} \vee x_9$$
$$\varphi_7 = x_8 \vee \overline{x_7} \vee \overline{x_9}$$

$$x_1 \Rightarrow x_4 \; [\varphi_1]$$
$$x_3 \Rightarrow x_8 \; [\varphi_2],\; x_{12} \; [\varphi_3]$$
$$x_2 \Rightarrow x_{11} \; [\varphi_4]$$
$$x_7 \Rightarrow x_{13} \; [\varphi_5],\; x_9 \; [\varphi_6],\; \overline{x_9} \; [\varphi_7]$$

Illustration

$$\varphi_1 = x_1 \vee x_4$$
$$\varphi_2 = x_1 \vee \overline{x_3} \vee \overline{x_8}$$
$$\varphi_3 = x_1 \vee x_8 \vee x_{12}$$
$$\varphi_4 = x_2 \vee x_{11}$$
$$\varphi_5 = \overline{x_3} \vee \overline{x_7} \vee x_{13}$$
$$\varphi_6 = \overline{x_3} \vee \overline{x_7} \vee \overline{x_{13}} \vee x_9$$
$$\varphi_7 = x_8 \vee \overline{x_7} \vee \overline{x_9}$$

$$x_1 \Rightarrow x_4 \; [\varphi_1]$$
$$x_3 \Rightarrow x_8 \; [\varphi_2],\; x_{12} \; [\varphi_3]$$
$$x_2 \Rightarrow x_{11} \; [\varphi_4]$$
$$x_7 \Rightarrow x_{13} \; [\varphi_5],\; x_9 \; [\varphi_6],\; \overline{x_9} \; [\varphi_7]$$

Illustration

$$\varphi_1 = x_1 \vee x_4$$
$$\varphi_2 = x_1 \vee \overline{x_3} \vee \overline{x_8}$$
$$\varphi_3 = x_1 \vee x_8 \vee x_{12}$$
$$\varphi_4 = x_2 \vee x_{11}$$
$$\varphi_5 = \overline{x_3} \vee \overline{x_7} \vee x_{13}$$
$$\varphi_6 = \overline{x_3} \vee \overline{x_7} \vee \overline{x_{13}} \vee x_9$$
$$\varphi_7 = x_8 \vee \overline{x_7} \vee \overline{x_9}$$

$$x_1 \Rightarrow x_4\ [\varphi_1]$$
$$x_3 \Rightarrow x_8\ [\varphi_2],\ x_{12}\ [\varphi_3]$$
$$x_2 \Rightarrow x_{11}\ [\varphi_4]$$
$$x_7 \Rightarrow x_{13}\ [\varphi_5],\ x_9\ [\varphi_6],\ \overline{x_9}\ [\varphi_7]$$

Illustration

$\varphi_1 = x_1 \vee x_4$
$\varphi_2 = x_1 \vee \overline{x_3} \vee \overline{x_8}$
$\varphi_3 = x_1 \vee x_8 \vee x_{12}$
$\varphi_4 = x_2 \vee x_{11}$
$\varphi_5 = \overline{x_3} \vee \overline{x_7} \vee x_{13}$
$\varphi_6 = \overline{x_3} \vee \overline{x_7} \vee \overline{x_{13}} \vee x_9$
$\varphi_7 = x_8 \vee \overline{x_7} \vee \overline{x_9}$

$x_1 \Rightarrow x_4$ $[\varphi_1]$
$x_3 \Rightarrow x_8$ $[\varphi_2]$, x_{12} $[\varphi_3]$
$x_2 \Rightarrow x_{11}$ $[\varphi_4]$
$x_7 \Rightarrow x_{13}$ $[\varphi_5]$, x_9 $[\varphi_6]$, $\overline{x_9}$ $[\varphi_7]$

Contradiction because of $\overline{x_3}$, $\overline{x_7}$, x_8

Illustration

$\varphi_1 = x_1 \vee x_4$
$\varphi_2 = x_1 \vee \overline{x_3} \vee \overline{x_8}$
$\varphi_3 = x_1 \vee x_8 \vee x_{12}$
$\varphi_4 = x_2 \vee x_{11}$
$\varphi_5 = \overline{x_3} \vee \overline{x_7} \vee x_{13}$
$\varphi_6 = \overline{x_3} \vee \overline{x_7} \vee \overline{x_{13}} \vee x_9$
$\varphi_7 = x_8 \vee \overline{x_7} \vee \overline{x_9}$

$x_1 \Rightarrow x_4$ $[\varphi_1]$
$x_3 \Rightarrow x_8$ $[\varphi_2]$, x_{12} $[\varphi_3]$
$x_2 \Rightarrow x_{11}$ $[\varphi_4]$
$x_7 \Rightarrow x_{13}$ $[\varphi_5]$, x_9 $[\varphi_6]$, $\overline{x_9}$ $[\varphi_7]$

Contradiction because of $\overline{x_3}$, $\overline{x_7}$, x_8
$\beta = \overline{x_3} \vee \overline{x_7} \vee x_8$ Conflict memory

Limitations

Thou shalt calculate only booleans

What's a theory ?

Définition (Theory)

A theory is an set of symbols and rules specifying the meaning of those symbols and their grammar (how they can be combined together).

What's a theory good for ?

To solve $a + b \geq 3$, we need to know about :

- identify symbols 3, a and b as members of the same set ($\mathbf{R}$)
- specify the meaning of the symbol $+$ (what is a sum)
- specify the meaning of the symbol $\geq$ and deduce a constraint
- specify what is the sum of two reals
- a way to solve the equation

What's a theory good for ?

To solve $a + b \geq 3$, we need to know about :

- identify symbols 3, a and b as members of the same set ($\mathbf{R}$)
- specify the meaning of the symbol $+$ (what is a sum)
- specify the meaning of the symbol $\geq$ and deduce a constraint
- specify what is the sum of two reals
- a way to solve the equation

Real arithmetic theory gives us the necessary tools :

What's a theory good for ?

To solve $a + b \geq 3$, we need to know about :

- identify symbols 3, a and b as members of the same set ($\mathbf{R}$)
- specify the meaning of the symbol $+$ (what is a sum)
- specify the meaning of the symbol $\geq$ and deduce a constraint
- specify what is the sum of two reals
- a way to solve the equation

Real arithmetic theory gives us the necessary tools :

- $(\mathbf{R}, +, *)$ as a set with properties
- evaluations rules

What's a theory good for ?

To solve $a + b \geq 3$, we need to know about :

- identify symbols 3, a and b as members of the same set ($\mathbf{R}$)
- specify the meaning of the symbol $+$ (what is a sum)
- specify the meaning of the symbol $\geq$ and deduce a constraint
- specify what is the sum of two reals
- a way to solve the equation

Real arithmetic theory gives us the necessary tools :

- $(\mathbf{R}, +, *)$ as a set with properties
- evaluations rules

Mature solvers : Linear Programming, simplex algorithm, etc.

How to make out theories with SAT ?

1. Reduce the theory-formula into a SAT formula by introducing variables
2. Find a conjunction of litterals using SAT solvers
3. Pass this conjunction to a solver modulo theory
4. Propagates given results as constraints via equalities

Illustration

Let the formula $((a = 1) \vee (a = 2)) \wedge (a \geq 3 \wedge ((b \leq 2) \vee (b \geq 3))$

Illustration

Let the formula $((a = 1) \vee (a = 2)) \wedge (a \geq 3 \wedge ((b \leq 2) \vee (b \geq 3))$
There is logic AND arithmetic
Are there reals making this formula true ?

Illustration

Let the formula $((a = 1) \vee (a = 2)) \wedge (a \geq 3 \wedge ((b \leq 2) \vee (b \geq 3))$
There is logic AND arithmetic
Are there reals making this formula true ?
Comment faire ? Créer des variables et les passer à SAT. Par exemple :
$x_1 : a = 1, x_2 : a = 2, x3 : a \geq 3, x_4 : b \leq 2, x5 : b \geq 3$

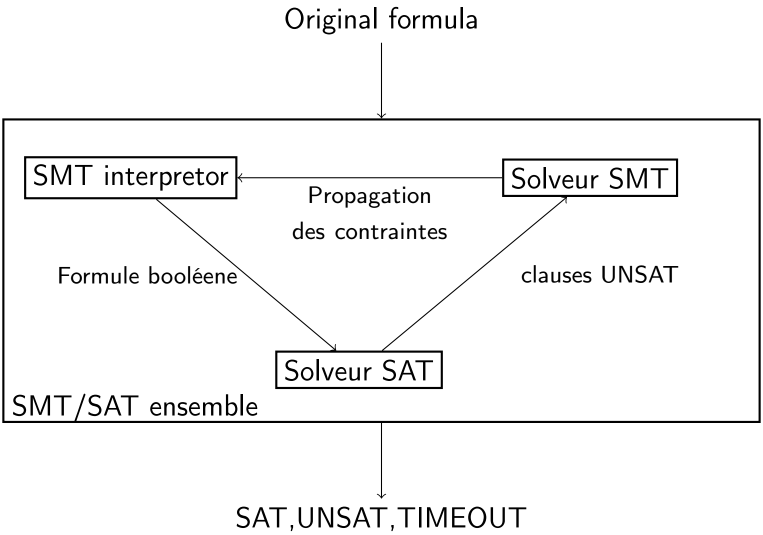
Illustration

Let the formula $((a = 1) \vee (a = 2)) \wedge (a \geq 3 \wedge ((b \leq 2) \vee (b \geq 3))$
There is logic AND arithmetic
Are there reals making this formula true ?
Comment faire ? Créer des variables et les passer à SAT. Par exemple :
$x_1 : a = 1, x_2 : a = 2, x3 : a \geq 3, x_4 : b \leq 2, x5 : b \geq 3$
On obtient alors : $(x_1 \vee x_2) \wedge (x_3 \wedge (x_4 \vee x_5))$

Illustration

Let the formula $((a = 1) \vee (a = 2)) \wedge (a \geq 3 \wedge ((b \leq 2) \vee (b \geq 3))$
There is logic AND arithmetic
Are there reals making this formula true ?
Comment faire ? Créer des variables et les passer à SAT. Par exemple :
$x_1 : a = 1, x_2 : a = 2, x3 : a \geq 3, x_4 : b \leq 2, x5 : b \geq 3$
On obtient alors : $(x_1 \vee x_2) \wedge (x_3 \wedge (x_4 \vee x_5))$
It's a SAT problem !

Illustration



Application concrète

Logiciels : Z3, CVC4, Yices, Simplify, Alt-Ergo

Que fournir en entrée ?

Déclarer des variables d'entrées (fonctions muettes) contraintes sous forme d'inégalités linéaires (ou affines) spécifier le flot de contrôle axiomes éventuels (définitions de fonctions) propriétés à vérifier

Exemple : identité sur un réseau jouet

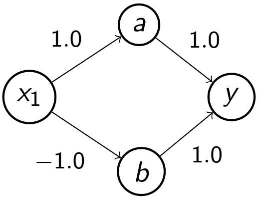


Figure – Pour $x_1 \geq 0$, on a l'identité

```
(set-logic QF_LRA)

;; Declare the neuron variables

(declare-fun x1 () Real)
(declare-fun a () Real)
(declare-fun b () Real)
(declare-fun y () Real)

;; Bound input ranges

(assert (>= x1 0))

;; Layer 1

(assert (let ((ws (* x1 1.0)))
(= a (ite (>= ws 0) ws 0))))
(assert (let ((ws (* x1 (- 1.0))))
(= b (ite (>= ws 0) ws 0))))

;; Layer 2
(assert (let ((ws (+ (* a 1.0) (* b  1.0))))
(= y ws)))

;; to check
(assert (= y x1))
(check-sat)
```

Exemple : identité sur un réseau jouet

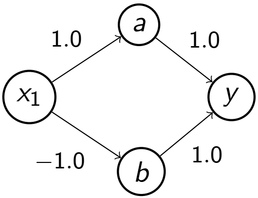


Figure – Pour $x_1 \geq 0$, on a l'identité

```
(set-logic QF_LRA)

;; Declare the neuron variables

(declare-fun x1 () Real)
(declare-fun a () Real)
(declare-fun b () Real)
(declare-fun y () Real)

;; Bound input ranges

(assert (>= x1 0))

;; Layer 1

(assert (let ((ws (* x1 1.0)))
(= a (ite (>= ws 0) ws 0))))
(assert (let ((ws (* x1 (- 1.0))))
(= b (ite (>= ws 0) ws 0))))

;; Layer 2
(assert (let ((ws (+ (* a 1.0) (* b  1.0))))
(= y ws)))

;; to check
(assert (= y x1))
(check-sat)
```

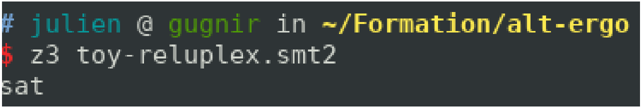


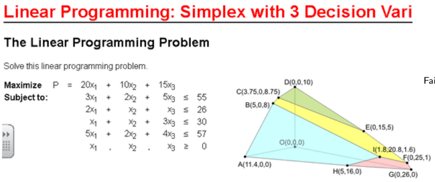
Figure – Formule satisfaite

Articles

- Towards Fast Computation of Certified Robustness for ReLU Networks, Tsui-Wei et al, 2018
- Reluplex : An Efficient SMT Solver for Verifying Deep Neural Networks, Katz et al, 2017
- DeepSafe : A Data-driven Approach for Assessing Robustness of Neural Networks, Gopinath et al, 2018

ReLuPlex : Simplexe + ReLu

Simplex algorithm : for a set of affine constraints, find the optimal solution. If it exists, the solution is at an edge of the constraint polytope


Linear Programming: Simplex with 3 Decision Vari

The Linear Programming Problem

Solve this linear programming problem.

Maximize $P = 20x_1 + 10x_2 + 15x_3$

Subject to:

$3x_1 + 2x_2 + 5x_3 \le 55$

$2x_1 + x_2 + x_3 \le 26$

$x_1 + x_2 + 3x_3 \le 30$

$5x_1 + 2x_2 + 4x_3 \le 57$

$x_1 , x_2 , x_3 \ge 0$

Implemented as an array with update rules

$$\text{Pivot}_1 \quad \frac{x_i \in \mathcal{B}, \quad \alpha(x_i) < l(x_i), \quad x_j \in \text{slack}^+(x_i)}{T := pivot(T,i,j), \quad \mathcal{B} := \mathcal{B} \cup \{x_j\} \setminus \{x_i\}}$$

$$\text{Pivot}_2 \quad \frac{x_i \in \mathcal{B}, \quad \alpha(x_i) > u(x_i), \quad x_j \in \text{slack}^-(x_i)}{T := pivot(T,i,j), \quad \mathcal{B} := \mathcal{B} \cup \{x_j\} \setminus \{x_i\}}$$

$$\text{Update} \quad \frac{x_j \notin \mathcal{B}, \quad \alpha(x_j) < l(x_j) \vee \alpha(x_j) > u(x_j), \quad l(x_j) \le \alpha(x_j) + \delta \le u(x_j)}{\alpha := update(\alpha, x_j, \delta)}$$

$$\text{Failure} \quad \frac{x_i \in \mathcal{B}, \quad (\alpha(x_i) < l(x_i) \ \wedge \ \text{slack}^+(x_i) = \emptyset) \vee (\alpha(x_i) > u(x_i) \ \wedge \ \text{slack}^-(x_i) = \emptyset)}{\texttt{UNSAT}}$$

$$\text{Success} \quad \frac{\forall x_i \in \mathcal{X}.\ l(x_i) \le \alpha(x_i) \le u(x_i)}{\texttt{SAT}}$$

ReLuPlex : Simplexe + ReLu

- Two variables for each ReLu : backward and forward
- Updates rules for ReLu inside of Simplex algorithm

$$\mathsf{Update}_b \quad \frac{x_i \notin \mathcal{B}, \quad \langle x_i, x_j \rangle \in R, \quad \alpha(x_j) \neq \max(0, \alpha(x_i)), \quad \alpha(x_j) \geq 0}{\alpha := update(\alpha, x_i, \alpha(x_j) - \alpha(x_i))}$$

$$\mathsf{Update}_f \quad \frac{x_j \notin \mathcal{B}, \quad \langle x_i, x_j \rangle \in R, \quad \alpha(x_j) \neq \max(0, \alpha(x_i))}{\alpha := update(\alpha, x_j, \max(0, \alpha(x_i)) - \alpha(x_j))}$$

$$\mathsf{PivotForRelu} \quad \frac{x_i \in \mathcal{B}, \quad \exists x_l.\ \langle x_i, x_l \rangle \in R \vee \langle x_l, x_i \rangle \in R, \quad x_j \notin \mathcal{B}, \quad T_{i,j} \neq 0}{T := pivot(T, i, j), \quad \mathcal{B} := \mathcal{B} \cup \{x_j\} \setminus \{x_i\}}$$

$$\mathsf{ReluSplit} \quad \frac{\langle x_i, x_j \rangle \in R, \quad l(x_i) < 0, \quad u(x_i) > 0}{u(x_i) := 0 \qquad l(x_i) := 0}$$

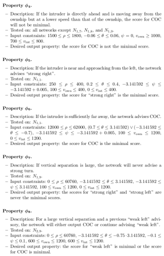
$$\mathsf{ReluSuccess} \quad \frac{\forall x \in \mathcal{X}.\ l(x) \leq \alpha(x) \leq u(x), \quad \forall \langle x^b, x^f \rangle \in R.\ \alpha(x^f) = \max(0, \alpha(x^b))}{\mathtt{SAT}}$$

DeepSafe : partition the input space

- Parition the input space using non-supervised clustering
- Uses SMT solvers to prove a given region robust regarding a certain label
- Partial robustness

Experimental setting

- ACAS Xu neural networks : Inputs are sensors informations (7 dimensions), output are instructions given to the pilot (5 dimensions)
- 6 layers, 7 or 9 neurons per layer, fully connected



Property ϕ_4.

- Description: If the intruder is directly ahead and is moving away from the ownship but at a lower speed than that of the ownship, the score for COC will not be minimal.
- Tested on: all networks except $N_{1,7}$, $N_{1,8}$, and $N_{1,9}$.
- Input constraints: $1500 \le \rho \le 1800$, $-0.06 \le \theta \le 0.06$, $\psi = 0$, $v_{\mathrm{own}} \ge 1000$, $700 \le v_{\mathrm{int}} \le 800$.
- Desired output property: the score for COC is not the minimal score.

Property ϕ_5.

- Description: If the intruder is near and approaching from the left, the network advises "strong right".
- Tested on: $N_{1,1}$.
- Input constraints: $250 \le \rho \le 400$, $0.2 \le \theta \le 0.4$, $-3.141592 \le \psi \le -3.141592 + 0.005$, $100 \le v_{\mathrm{own}} \le 400$, $0 \le v_{\mathrm{int}} \le 400$.
- Desired output property: the score for "strong right" is the minimal score.

Property ϕ_6.

- Description: If the intruder is sufficiently far away, the network advises COC.
- Tested on: $N_{1,1}$.
- Input constraints: $12000 \le \rho \le 62000$, $(0.7 \le \theta \le 3.141592) \vee (-3.141592 \le \theta \le -0.7)$, $-3.141592 \le \psi \le -3.141592 + 0.005$, $100 \le v_{\mathrm{own}} \le 1200$, $0 \le v_{\mathrm{int}} \le 1200$.
- Desired output property: the score for COC is the minimal score.

Property ϕ_7.

- Description: If vertical separation is large, the network will never advise a strong turn.
- Tested on: $N_{1,9}$.
- Input constraints: $0 \le \rho \le 60760$, $-3.141592 \le \theta \le 3.141592$, $-3.141592 \le \psi \le 3.141592$, $100 \le v_{\mathrm{own}} \le 1200$, $0 \le v_{\mathrm{int}} \le 1200$.
- Desired output property: the scores for "strong right" and "strong left" are never the minimal scores.

Property ϕ_8.

- Description: For a large vertical separation and a previous "weak left" advisory, the network will either output COC or continue advising "weak left".
- Tested on: $N_{2,9}$.
- Input constraints: $0 \le \rho \le 60760$, $-3.141592 \le \theta \le -0.75 \cdot 3.141592$, $-0.1 \le \psi \le 0.1$, $600 \le v_{\mathrm{own}} \le 1200$, $600 \le v_{\mathrm{int}} \le 1200$.
- Desired output property: the score for "weak left" is minimal or the score for COC is minimal.

Figure – Exemple of verified properties

Results : ReLuPlex

Property ϕ_4.

- Description: If the intruder is directly ahead and is moving away from the ownship but at a lower speed than that of the ownship, the score for COC will not be minimal.
- Tested on: all networks except $N_{1,7}$, $N_{1,8}$, and $N_{1,9}$.
- Input constraints: $1500 \leq \rho \leq 1800$, $-0.06 \leq \theta \leq 0.06$, $\psi = 0$, $v_{\text{own}} \geq 1000$, $700 \leq v_{\text{int}} \leq 800$.
- Desired output property: the score for COC is not the minimal score.

Property ϕ_5.

- Description: If the intruder is near and approaching from the left, the network advises "strong right".
- Tested on: $N_{1,1}$.
- Input constraints: $250 \leq \rho \leq 400$, $0.2 \leq \theta \leq 0.4$, $-3.141592 \leq \psi \leq -3.141592 + 0.005$, $100 \leq v_{\text{own}} \leq 400$, $0 \leq v_{\text{int}} \leq 400$.
- Desired output property: the score for "strong right" is the minimal score.

Property ϕ_6.

- Description: If the intruder is sufficiently far away, the network advises COC.
- Tested on: $N_{1,1}$.
- Input constraints: $12000 \leq \rho \leq 62000$, $(0.7 \leq \theta \leq 3.141592) \vee (-3.141592 \leq \theta \leq -0.7)$, $-3.141592 \leq \psi \leq -3.141592 + 0.005$, $100 \leq v_{\text{own}} \leq 1200$, $0 \leq v_{\text{int}} \leq 1200$.
- Desired output property: the score for COC is the minimal score.

Property ϕ_7.

- Description: If vertical separation is large, the network will never advise a strong turn.
- Tested on: $N_{1,9}$.
- Input constraints: $0 \leq \rho \leq 60760$, $-3.141592 \leq \theta \leq 3.141592$, $-3.141592 \leq \psi \leq 3.141592$, $100 \leq v_{\text{own}} \leq 1200$, $0 \leq v_{\text{int}} \leq 1200$.
- Desired output property: the scores for "strong right" and "strong left" are never the minimal scores.

Property ϕ_8.

- Description: For a large vertical separation and a previous "weak left" advisory, the network will either output COC or continue advising "weak left".
- Tested on: $N_{2,9}$.
- Input constraints: $0 \leq \rho \leq 60760$, $-3.141592 \leq \theta \leq -0.75 \cdot 3.141592$, $-0.1 \leq \psi \leq 0.1$, $600 \leq v_{\text{own}} \leq 1200$, $600 \leq v_{\text{int}} \leq 1200$.
- Desired output property: the score for "weak left" is minimal or the score for COC is minimal.

Figure – Exemple of verified properties

Table 2: Verifying properties of the ACAS Xu networks.

	Networks	Result	Time	Stack	Splits
ϕ_1	41	UNSAT	394517	47	1522384
	4	TIMEOUT			
ϕ_2	1	UNSAT	463	55	88388
	35	SAT	82419	44	284515
ϕ_3	42	UNSAT	28156	22	52080
ϕ_4	42	UNSAT	12475	21	23940
ϕ_5	1	UNSAT	19355	46	58914
ϕ_6	1	UNSAT	180288	50	548496
ϕ_7	1	TIMEOUT			
ϕ_8	1	SAT	40102	69	116697

Results : DeepSafe

MNIST proven robust for certain labels within 12 hours of testing, with 10 hours of clustering (80 clusters).

Questions ?

:)