Set-based methods in programs and systems verification

Sylvie Putot and Eric Goubault
Cosynus team, LIX, Ecole Polytechnique

SWIM 2016, ENS Lyon
Validation of programs (discrete systems)

- Does it crash? can we bound program variables? does it compute the square root? at what precision?
- For embedded systems, work on control code since the 2000s (FLUCTUAT, Astrée etc.) in particular using abstract interpretation (Cousot & Cousot 1977-) mostly on invariant properties
- Connections to set-based methods

```c
float xn, xnp1, residu, Input, Output, shouldbezero; int i = 0;
float eps = 0.00000001;
Input = FBETWEEN(16.0, 16.002);
xn = 1.0 / Input; xnp1 = xn;
residu = 2.0 * eps * (xn + xnp1) / (xn + xnp1);
while (fabs(residu) > eps) {
    xnp1 = xn * (1.875 + Input * xn * xn
              * (-1.25 + 0.375 * Input * xn * xn));
    residu = 2.0 * (xnp1 - xn) / (xn + xnp1);
    xn = xnp1;
    i++;
}
Output = 1.0 / xnp1;
shouldbezero = Output - sqrt(Input);
```

Validation of hybrid systems (discrete and continuous systems)

- For embedded systems control, many properties only provable on closed-loop systems (with the physical environment)
- Even more connections to set-based methods (Taylor models as in e.g. Berz & Makino ~1985)
- Connections with model-checking (Clarke, Emerson, Queille, Sifakis 1980) : generalisation of reachability and invariance properties (if some value is bounded for some time, can we reach some prescribed state?)
Contents of the talk

Quick guided tour on our work on abstract interpretation of numerical programs

- Set-based methods and Abstract Interpretation, for validation of programs (here, just in real numbers!)
  - An introduction to abstract interpretation
  - Zonotopes for reachability, invariant synthesis, and functional proofs of algorithms

Some work in progress on abstract interpretation of hybrid systems

- Extension of zonotopic methods: inner- and outer- approximations of the set of solutions of uncertain ODEs
- Use for temporal verification, LTL, CTL and (abstract) model-checking
The overall context, in the last 15 years or so

Computation of sets of reachable values of variables at any program points (FLUCTUAT)

- Need to bound real and finite precision values of variables, and the difference between them, decomposed on the provenance of these errors
- Accurate outer-approximation with affine forms
- Implemented in the FLUCTUAT analyzer for C programs

But how pessimistic are the results?

- Joint use of inner- and outer-approximations to characterize the quality of analysis results
  - Inner-approximation: sets of values of the outputs, that are sure to be reached for some inputs in the specified ranges.
  - Use of affine forms with generalized intervals as coefficients
Static analysis by abstract interpretation (Cousot 77)

**Basics**

- Choose properties of interest (for instance values of variables)
- Outer-approximate them in an abstract lattice (partially ordered structure with least upper bounds/greatest lower bounds) whose elements are particular “tractable” sets
- Interpret computations in this lattice

---

<table>
<thead>
<tr>
<th>Intervals</th>
<th>Octogons</th>
<th>Polyhedrons</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="intervals.png" alt="" /></td>
<td><img src="octagons.png" alt="" /></td>
<td><img src="polyhedrons.png" alt="" /></td>
</tr>
</tbody>
</table>
Abstract semantics of programs in intervals (for invariant generation)

- Program seen as a **discrete dynamical system** $X^{n+1} = F(X^n)$
  - based on a notion of control points in the program
  - equations describe how values of variables are collected at each control point, for all possible executions (collecting semantics)

```c
void main() {
    int x=[-100,50]; [1]
    while [2] (x < 100) {
        [3] x=x+1; [4]
    } [5]
}
```

\[
\begin{align*}
    x_0 &= \top \\
    x_1 &= [-100, 50] \\
    x_2 &= x_1 \cup x_4 \\
    x_3 &= ]-\infty, 99] \cap x_2 \\
    x_4 &= x_3 + [1, 1] \\
    x_5 &= [100, +\infty] \cap x_2
\end{align*}
\]

**Invariants generation = least fixed point computation**

- The sets of possible values of variables at control points are invariants of $F$, computed as the **least fixpoint of the system**
- $F$ monotonic on a complete lattice, least fixpoint exists
Invariants and validation

- Invariants allow to conclude about the safety (for instance absence of run-time errors)
- E.g. we will find for:
  ```
  int x=[-100,50]; [1]
  x₃ = [-100, 99] and the program will not run into an overflow
  ```

Computation of invariants as the least fixpoint \( X = F(X) \)

- Limit of the Kleene iteration (Jacobi/Gauss-Seidel like method)
  \( X^0 = ⊥, X^1 = F(X^0), \ldots, X^{k+1} = X^k \cup F(X^k) \)
  - with convergence acceleration to terminate in finite time
- An alternative: policy iteration (Newton-like method)

The least fixpoint is the best inductive invariant \((F(X) \subseteq X)\) ...

- ... but invariants are not always inductive (in a given abstract domain)
- Search for a disjunction which is inductive: algorithm inspired from constraint programming (Mine and al. 2015 on boxes, B. Kabi’s talk)
Affine Arithmetic (Comba & Stolfi 93) for real-numbers abstraction

Affine forms

- Affine form for variable $x$:
  \[
  \hat{x} = x_0 + x_1 \varepsilon_1 + \ldots + x_n \varepsilon_n, \; x_i \in \mathbb{R}
  \]

  where the $\varepsilon_i$ are symbolic variables (\textit{noise symbols}), with value in $[-1, 1]$.

- Sharing $\varepsilon_i$ between variables expresses implicit dependency

- Interval concretization of affine form $\hat{x}$:
  \[
  \left[ x_0 - \sum_{i=0}^{n} |x_i|, x_0 + \sum_{i=0}^{n} |x_i| \right] = x_0 + [-\| (x_i) \|_1, \| (x_i) \|_1]
  \]

Geometric concretization as zonotopes (center symmetric polytopes)

\[
\begin{align*}
\hat{x} &= 20 - 4\varepsilon_1 + 2\varepsilon_3 + 3\varepsilon_4 \\
\hat{y} &= 10 - 2\varepsilon_1 + \varepsilon_2 - \varepsilon_4
\end{align*}
\]

Huge literature - (dual) generator representation of a polytope!
Affine arithmetic

Basic arithmetic operations

- **Assignment** $x := [a, b]$ introduces a noise symbol:
  $$\hat{x} = \frac{(a + b)}{2} + \frac{(b - a)}{2} \varepsilon_i.$$

- **Addition/subtraction** are exact:
  $$\hat{x} + \hat{y} = (x_0 + y_0) + (x_1 + y_1)\varepsilon_1 + \ldots + (x_n + y_n)\varepsilon_n$$

- **Non linear operations**: approximate linear form, new noise term bounding the approximation error
  $$\hat{x} \times \hat{y} = x_0 y_0 + \sum_{i=0}^{n} (x_0 y_i + x_i y_0)\varepsilon_i + \left( \sum_{1 \leq i \neq j \leq n} |x_i y_j| \right)\varepsilon_{n+1}$$
  (better formulas including SDP computations of the new term)

- Close to Taylor models of order 1: **low time complexity!** and easy to implement on a finite-precision machine
Set-theoretic operations

**Reminder!**

Need to define an order relation and interpret set-theoretic operations such as \( \cup \) and \( \cap \) (or at least outer-approximations), as for e.g.:

\[
\begin{align*}
  x_1 &= [-100, 50] \\
  x_2 &= x_1 \cup x_4 \\
  x_3 &= ] - \infty, 99] \cap x_2 \\
  x_4 &= x_3 + [1, 1] \\
  x_5 &= [100, +\infty[ \cap x_2
\end{align*}
\]

**Note**

- We are actually abstracting input-output relationships, not just the image of functions (see Arxiv 2008 & 2009 & FMSD 2016)
- This is the major difference with respect to classical work on zonotopes (Comba & Stolfi, Girard etc.)
A simple example: functional interpretation

\[ \text{real } x = [0, 10]; \]
\[ \text{real } y = x^2 - x; \]

\[ \hat{x} \]
\[ \hat{y} \]

Abstraction of \( x \): \( x = 5 + 5\varepsilon_1 \)

Abstraction of function \( x \to y = x^2 - x \) as

\[ y = 32.5 + 50\varepsilon_1 + 12.5\eta_1 \]
A simple example: functional interpretation

\begin{align*}
\text{real } x &= [0,10]; \\
\text{real } y &= x^2 - x;
\end{align*}

Abstraction of \( x \): \( x = 5 + 5\varepsilon_1 \)

Abstraction of function \( x \to y = x^2 - x \) as

\begin{align*}
y &= 32.5 + 50\varepsilon_1 + 12.5\eta_1 \\
   &= -17.5 + 10x + 12.5\eta_1
\end{align*}
Set operations on affine sets / zonotopes: meet

Test interpretation: intersection of zonotopes with guards are not zonotopes!

Interpreting tests

- Translate the condition on noise symbols
- Abstract domain for the noise symbols: intervals, octagons, etc.
- Test interpretation is (interval) constraint propagation

Example

real x = [0,10]; real y = 2*x;
if (y >= 10) y = x;

- Affine forms before tests: $x = 5 + 5\varepsilon_1$, $y = 10 + 10\varepsilon_1$
- In the if branch: constraint $\varepsilon_1 \geq 0$
Test interpretation: intersection of zonotopes with guards are not zonotopes!

Interpreting tests
- Translate the condition on noise symbols
- Abstract domain for the noise symbols: intervals, octagons, etc.
- Test interpretation is (interval) constraint propagation

When going to finite precision analysis
- Constraints give condition on idealized (in reals) and machine (in floating-point numbers) paths to be taken by an execution
- Unstable test detection is a constraint satisfaction problem
Join operator

\[
\begin{align*}
(\hat{x} &= 3 + \varepsilon_1 + 2\varepsilon_2, \hat{u} = 0 + \varepsilon_1 + \varepsilon_2) \cup (\hat{y} = 1 - 2\varepsilon_1 + \varepsilon_2, \hat{u} = 0 + \varepsilon_1 + \varepsilon_2) = \\
(\hat{x} \cup \hat{y} &= 2 + \varepsilon_2 + 3\eta_1, \\
\hat{u} &= 0 + \varepsilon_1 + \varepsilon_2)
\end{align*}
\]

Construction (low complexity!: \(O(n \times p)\))

- Keep “minimal common dependencies”

\[
z_i = \arg\min_{x_i \land y_i \leq r \leq x_i \lor y_i} |r|, \forall i \geq 1
\]
Join operator

\[
\begin{align*}
\hat{x} &= 3 + \varepsilon_1 + 2\varepsilon_2 \\
\hat{u} &= 0 + \varepsilon_1 + \varepsilon_2
\end{align*}
\]

∪

\[
\begin{align*}
\hat{y} &= 1 - 2\varepsilon_1 + \varepsilon_2 \\
\hat{u} &= 0 + \varepsilon_1 + \varepsilon_2
\end{align*}
\]

= 

\[
\begin{align*}
\hat{x} \cup \hat{y} &= 2 + \varepsilon_2 + 3\eta_1 \\
\hat{u} &= 0 + \varepsilon_1 + \varepsilon_2
\end{align*}
\]

Construction (low complexity!: \(\mathcal{O}(n \times p)\))

- Keep “minimal common dependencies”

\[
z_i = \arg\min_{x_i \wedge y_i \leq r \leq x_i \vee y_i} |r|, \quad \forall i \geq 1
\]

- For each dimension, concretization is the interval union of the concretizations: \(\gamma(\hat{x} \cup \hat{y}) = \gamma(\hat{x}) \cup \gamma(\hat{y})\)

- A more precise upper bound: NSAD 2012
Convergence of fixpoint computation: from concrete to abstract

General result on recursive linear filters, pervasive in embedded programs:

\[ x_{k+n+1} = \sum_{i=1}^{n} a_i x_{k+i} + \sum_{j=1}^{n+1} b_j e_{k+j}, \quad e_l \in [m, M] \]

- Concrete scheme has **bounded outputs** iff zeros of \( x^n - \sum_{i=0}^{n-1} a_{i+1} x^i \) have modulus strictly lower than 1.
- Then our Kleene iteration (with some initial unfolding \( p \) and uncyclic unfolding \( q \)) converges towards a **finite outer-approximation** of the outputs

\[ \hat{X}_i = \hat{X}_{i-1} \cup F^q(E_i, \ldots, E_{i-k}, \hat{X}_{i-1}, \ldots, \hat{X}_{i-k}) \]

in finite time

- The abstract scheme is a perturbation (by the join operation) of the concrete scheme
- Proof uses: for each dimension \( \gamma(\hat{x} \cup \hat{y}) = \gamma(\hat{x}) \cup \gamma(\hat{y}) \) and \( F^q \) is contracting “enough” for some \( q \)

Generalization to some recurrent polynomial schemes
\[ S_{n+2} = 0.7E_{n+2} - 1.3E_{n+1} + 1.1E_n + 1.4S_{n+1} - 0.7S_n \]

- after initial unfolding (10)
- first cyclic unfolding (80) - first join
- after first join, perturbation of the original numerical scheme
- then second cyclic unfolding, contracting back: second join and post-fixpoint
Illustration: a simple order 2 filter

\[ S_{n+2} = 0.7E_{n+2} - 1.3E_{n+1} + 1.1E_n + 1.4S_{n+1} - 0.7S_n \]

Fixpoint (polyhedral outer-approximation of the ellipsoidal invariant) below:
Fluctuat: also properties of finite precision arithmetic (here Householder)

```c
#include "daed_builtins.h"
#include <math.h>
#define _EPS 0.00000001 /* 10^-8 */
int main ()
{
    float xn, xnp1, residu, Input, Output,
         should_be_zero;
    int i;
    Input = FBETWEEN(16.0, 16.002);
    xn = 1.0/Input; xnp1 = xn;
    residu = 2.0*_EPS*(xn+xnp1)/(xn+xnp1);
    i = 0;
    while (fabs(residu) > _EPS)
    {
        xnp1 = xn * (1.875 + Input*xn*xn*(-1.25+0.375*Input*xn*xn));
        residu = 2.0*(xnp1-xn)/(xn+xnp1);
        xn = xnp1;
        i++;
    }
    Output = 1.0 / xnp1;
    should_be_zero = Output - sqrt(Input);
    return 0;
}
```

Variables / Files
- Input (float)
- Output (float)
- i (integer)
- main (integer)
- residu (float)
- should_be_zero (float)
- signgam (integer)

Variable Interval
- Float:
  - Lower: -1.18123876e-6
  - Upper: 1.18123956e-6
- Real:
  - Lower: -1.02630258e-8
  - Upper: 1.02636675e-8
- Global error:
  - Lower: -1.17097598e-6
  - Upper: 1.17097576e-6
- Relative error:
  - Lower: -oo
  - Upper: +oo
- Higher Order error:
  - Lower: 0
  - Upper: 0
- At current point (17) *
  - Lower: -9.17837e-07
  - Upper: 9.17837e-07

Last analysis: 0.42 sec / 28672 Kilo Bytes
Inner-approximations?

Starting point
- Assert the quality of abstractions by looking at inner and outer approx
- Falsify properties
- Provide lower/upper bounds for convergence of numerical schemes

Newton algorithm for $a \rightarrow 1/a$, stop when $|x_{n+1} - x_n| < 5 \times 10^{-4}$

- **Outer approximation**: stopping criterion always satisfied after 4 iter ($|x(4) - x(3)| \subseteq [-2.6 \times 10^{-4}, 2.6 \times 10^{-4}]$).
- **Inner approximation**: some inputs falsify criterion on first 3 iterations ($[-7.7 \times 10^{-4}, -4.1 \times 10^{-4}] \subseteq x(3) - x(2)$).
How to do this?

Inner approximation much more difficult than outer approximation (some pointers)

- Linear case for ODEs [Kurzhanski-Varaiya HSCC 2000, Althoff et al. CDC 2007, Kanade et al. CAV 2009]
- Simulation-based local inner approximations [Ngkiem et al. HSCC 2010]
- Box bisections [Goldsztejn-Jaulin Reliable Computing 2010, Mullier-Goubault-Kieffer-Putot RC 2013]
- Parallelepipeds [Goldsztejn-Hayes SCAN 2006]
- Order 0 generalized affine forms [Goubault-Putot SAS 2007], order 1 [Goubault-Kieffer-Mullier-Putot HSCC 2014]

Will be useful in the rest of the talk...

- Check general temporal properties (TCTL in particular) on hybrid systems
- By a combination of inner- and outer- approximations
Generalized intervals

General bounds \( K = \{ [a, b], \ a \in \mathbb{R}, b \in \mathbb{R} \} \) (“improper” if \( a > b \))

Kaucher arithmetic (only when no dependencies between arguments!)

All proper intervals: outer-approx \((\forall x \in [x]) (\exists z \in [z]) (f(x) = z)\)
All improper intervals: inner-approx \((\forall z \in \text{pro} [z]) (\exists x \in \text{pro} [x])(f(x) = z)\).

Remedy: Mean-value theorem (à la Goldsztejn 2005) + affine arithmetic

Let \( f : \mathbb{R}^n \to \mathbb{R} \) differentiable, \((t_1, \ldots, t_n)\) a point in \([-1, 1]^n\) and \( \Delta_i \) such that

\[
\left\{ \frac{\partial f}{\partial \varepsilon_i}(\varepsilon_1, \ldots, \varepsilon_i, t_{i+1}, \ldots, t_n), \varepsilon_i \in [-1, 1] \right\} \subseteq \Delta_i.
\]

Then \( \tilde{f}(\varepsilon_1, \ldots, \varepsilon_n) = f(t_1, \ldots, t_n) + \sum_{i=1}^{n} \Delta_i(\varepsilon_i - t_i) \), means

- if \( \tilde{f}(\varepsilon_1^*, \ldots, \varepsilon_n^*), \varepsilon_i^* = [1, -1] \), computed with Kaucher arithmetic, is an improper interval, then \( \text{pro} \tilde{f}(\varepsilon_1^*, \ldots, \varepsilon_n^*) \) is an inner-approx of \( f(\varepsilon_1, \ldots, \varepsilon_n) \).
- \( \tilde{f}(\varepsilon_1, \ldots, \varepsilon_n), \varepsilon_i = [-1, 1] \), is an outer-approx of \( f(\varepsilon_1, \ldots, \varepsilon_n) \).
Generalized affine forms and inner range computation

Generalized affine forms

- The generalized mean-value theorem defines generalized affine forms: for \( f : \mathbb{R}^n \rightarrow \mathbb{R}, \)

\[
    f^\varepsilon(t_1, \ldots, t_n) + \sum_{i=1}^{n} \Delta_i(\varepsilon_i - t_i),
\]

where \( \left\{ \frac{\partial f^\varepsilon}{\partial \varepsilon_i}(\varepsilon), \varepsilon \in [-1, 1]^n \right\} \subseteq \Delta_i. \)

- We want an inductive computation of these forms on arithmetic expressions.

Order 0 forms

- The partial derivatives \( \Delta_i \) are evaluated with intervals.
- Example: \( f(x) = x^2 - x, x \in [2, 3], \) thus

\[
    f^\varepsilon(\varepsilon_1) = (2.5 + 0.5\varepsilon_1)^2 - (2.5 + 0.5\varepsilon_1).
\]

We get \( \tilde{f}^\varepsilon(\varepsilon_1) = 3.75 + [1.5, 2.5]\varepsilon_1, \) that can be interpreted as:

\[
    pro(3.75 + [1.5, 2.5][1, -1]) \subseteq f([-1, 1]) \subseteq 3.75 + [1.5, 2.5][-1, 1]
\]
Generalized affine forms

- The generalized mean-value theorem defines generalized affine forms: for \( f : \mathbb{R}^n \rightarrow \mathbb{R} \),

\[
f^\varepsilon(t_1, \ldots, t_n) + \sum_{i=1}^{n} \Delta_i(\varepsilon_i - t_i),
\]

where \( \left\{ \frac{\partial f^\varepsilon}{\partial \varepsilon_i}(\varepsilon), \varepsilon \in [-1, 1]^n \right\} \subseteq \Delta_i \).

- We want an inductive computation of these forms on arithmetic expressions

Order 0 forms

- The partial derivatives \( \Delta_i \) are evaluated with intervals

- Example: \( f(x) = x^2 - x \), \( x \in [2, 3] \), thus

\[
f^\varepsilon(\varepsilon_1) = (2.5 + 0.5\varepsilon_1)^2 - (2.5 + 0.5\varepsilon_1).
\]

We get \( \tilde{f}^\varepsilon(\varepsilon_1) = 3.75 + [1.5, 2.5]\varepsilon_1 \), that can be interpreted as:

\[
pro(3.75 + [1.5, -1.5]) \subseteq f([-1, 1]) \subseteq 3.75 + [-2.5, 2.5]
\]
Generalized affine forms

The generalized mean-value theorem defines generalized affine forms: for
\( f : \mathbb{R}^n \to \mathbb{R}, \)
\[
f^\varepsilon(t_1, \ldots, t_n) + \sum_{i=1}^{n} \Delta_i(\varepsilon_i - t_i),
\]
where \( \left\{ \frac{\partial f^\varepsilon}{\partial \varepsilon_i}(\varepsilon), \varepsilon \in [-1, 1]^n \right\} \subseteq \Delta_i. \)

We want an inductive computation of these forms on arithmetic expressions

Order 0 forms

The partial derivatives \( \Delta_i \) are evaluated with intervals

Example: \( f(x) = x^2 - x, x \in [2, 3], \) thus
\[
f^\varepsilon(\varepsilon_1) = (2.5 + 0.5\varepsilon_1)^2 - (2.5 + 0.5\varepsilon_1).
\]
We get \( \tilde{f}^\varepsilon(\varepsilon_1) = 3.75 + [1.5, 2.5]\varepsilon_1, \) that can be interpreted as:
\[
\text{pro}([5.25, 4.25]) \subseteq f([-1, 1]) \subseteq [1.25, 6.25]
\]
Generalized affine forms and inner range computation

**Generalized affine forms**

- The generalized mean-value theorem defines generalized affine forms: for \( f : \mathbb{R}^n \rightarrow \mathbb{R} \),

\[
f^\varepsilon(t_1, \ldots, t_n) + \sum_{i=1}^{n} \Delta_i(\varepsilon_i - t_i),
\]

where \( \left\{ \frac{\partial f^\varepsilon}{\partial \varepsilon_i}(\varepsilon), \ \varepsilon \in [-1, 1]^n \right\} \subseteq \Delta_i \).

- We want an inductive computation of these forms on arithmetic expressions

**Order 0 forms**

- The partial derivatives \( \Delta_i \) are evaluated with intervals

- Example: \( f(x) = x^2 - x, \ x \in [2, 3] \), thus

\[
f^\varepsilon(\varepsilon_1) = (2.5 + 0.5\varepsilon_1)^2 - (2.5 + 0.5\varepsilon_1).
\]

We get \( \tilde{f}^\varepsilon(\varepsilon_1) = 3.75 + [1.5, 2.5]\varepsilon_1 \), that can be interpreted as:

\[
[4.25, 5.25] \subseteq f([-1, 1]) \subseteq [1.25, 6.25]
\]

- Solves the single-occurrence limitation but not quite the dependency problem
Generalized affine forms

- The generalized mean-value theorem defines generalized affine forms: for \( f : \mathbb{R}^n \rightarrow \mathbb{R} \),
  \[
  f^\varepsilon(t_1, \ldots, t_n) + \sum_{i=1}^{n} \Delta_i(\varepsilon_i - t_i),
  \]
  where \( \left\{ \frac{\partial f^\varepsilon}{\partial \varepsilon_i}(\varepsilon), \quad \varepsilon \in [-1, 1]^n \right\} \subseteq \Delta_i \).

- We want an inductive computation of these forms on arithmetic expressions.

Order 1 forms

- Inductive computations with zonotopic outer-approximations of quantities and partial derivatives \( \Delta_i \): more precise that order 0.
- When computing the inner range of a scalar function as above, we use only the interval range \( \Delta_i \).
- But in general we have \( f : \mathbb{R}^n \rightarrow \mathbb{R}^p \) and thus vectors of generalized affine forms.
- Order 1 forms code some dependency between the components of \( f \) or \( f^\varepsilon \): also allows us to define joint inner range.
Algorithm to compute a set of boxes proved to be in the image of $f$:
- Based on input set bisection + a sufficient condition for a box $\tilde{y}$ to be in range($f, x$).
- Only needs an outer approximation of the Jacobian of $f$
- Goldzstejn-Jaulin 2010 ($f : \mathbb{R}^n \rightarrow \mathbb{R}^n$), MGKP 2013 (extension $f : \mathbb{R}^n \rightarrow \mathbb{R}^p$)

$$f_S(x) = \{ f(x) : x \in x \}$$

$$\tilde{x} + \Gamma(J, \tilde{x} - \tilde{x}, \tilde{y} - f(\tilde{x}))$$

$$f_S(\tilde{x})$$
Characterization of the joint inner range of order 1 affine vectors: example

Example

Let \( x = (x_1, x_2) \in [2, 3] \times [3, 4] \) and

\[
  f(x) = \begin{pmatrix}
  x_1^3 - 2x_1x_2 \\
  x_2^3 - 2x_1x_2
  \end{pmatrix}
\]

Joint inner range of the corresponding order 1 affine vectors costly but rarely needed
As an extension of classical program (discrete-time) analysis

- **Classical program analysis**: inputs given in ranges, possibly with bounds on the gradient between two values
  - Behaviour is often not realistic

- **Hybrid systems analysis**: analyze both physical environment and control software for better precision
  - Environment modelled by switched ODE systems
    - abstraction by guaranteed integration (the solver is guaranteed to outer-approximate the real solution)
  - Interaction between program and environment modelled by assertions in the program
    - sensor reads a variable value at time $t$ from the environment,
    - actuator sends a variable value at time $t$ to the environment,

- Other possible use of guaranteed integration in program analysis: **bound method error** of ODE solvers
Example: the ATV escape mechanism

```c
int main() {
    float ac[3];
    float x_nav[7], x_est[7];
    float x_interm[7];
    for(j=0; j<7; j++) {
        x_nav[j] = HYBRID_DVALUE("sensor", 0, j);
        RK4 (x_interm, x_nav, 0.075);
        RK4 (x_pred, x_interm, 0.925);
        estim(x_est, x_nav, x_pred);
        command(ac, x_est);
        HYBRID_PARAM("sensor", 0, ac[0], j);
    }
    // file sensor.h: EDO definition
    y_0 = -y_1 * (y_4 + w_12) - y_2 * (y_5 + w_22) - y_3 * (y_6 + w_32)
    y_1 =  y_0 * (y_4 + w_12) + y_2 * (y_6 + w_32) - y_3 * (y_5 + w_22)
    y_2 =  y_0 * (y_1 + w_22) + y_3 * (y_4 + w_12) - y_1 * (y_6 + w_32)
    y_3 =  y_0 * (y_6 + w_32) + y_1 * (y_5 + w_22) - y_2 * (y_4 + w_12)
    y_4 =  -y_5 * y_6 * i_1 + a_0
    y_5 =  -y_4 * y_6 * i_2 + a_1
    y_6 =  -y_4 * y_5 * i_3 + a_2
}
```

- Time is controlled by the program ($j$)
- Program changes parameters (HYBRID_PARAM: actuators) or mode (not here) of the ODE system
- Program reads from the environment (HYBRID_DVALUE: sensors) by calling the ODE guaranteed solver

Could demonstrate convergence towards the safe escape state (CAV 2009, DASIA 2009 with Olivier Bouissou).
Temporal logics: general properties on trajectories

Linear time temporal logics
- Simple LTL (Pnueli 1979)
- Metric temporal logics (Koymans 1990) used for falsification of properties in real-time (hybrid) systems, (see e.g. Sankaranarayanan & Fainekos, HSCC 2012)

Modalities
- 1st order predicate logics $\land, \lor, \Rightarrow, \neg$ etc.
- Temporal modalities; e.g.
  - $\Box p$: $p$ is true always in the future
  - $\Diamond p$: $p$ is going to be true at some point in the future

Adding time (MTL)
- We add a time interval as index to the modalities
- $\Box_{[0,300]} p$: $p$ is always true for times (in the future) between 0 and 300
- $\Diamond_{[5,10]} p$: $p$ is eventually true between times 5 and 10
Examples

This is a logics on states, for all trajectories

\[ \Diamond_{[0,2]} (y > 0.7) \text{ is true but } \Box_{[0,2]} (y > 0.7) \text{ is false} \]

Examples of interesting properties in control

- Stability (invariance) : \[ \Box_{[t,\infty]} (x \in K) \]
- A signal \( x_2 \) being close to a reference signal \( x_1 \):
  \[ \Box_{[0,\infty]} ((x_1 > 0.7) \Rightarrow \Diamond_{[0,0.1]} (x_2 > 0.7)) \]
  (whenever \( x_1 \) crosses threshold 0.7, so does \( x_2 \) within \([0, 0.1]\) time unit - think of a decision taken by a system computing in finite precision)
- Bounded-time stabilization with respect to disturbances: whenever signal gets outside its reference range, it should be brought in this range in bounded time and remain there for some time
  \[ \Box_{[0,\infty]} ((x \leq 100) \land ((x > 10) \Rightarrow \Diamond_{[0,150]} \Box_{[0,10]} (x \leq 10))) \]
Quantifying on executions

Branching time logics

- CTL, CTL* (Clarke, Emerson 1981)
- Add quantification over paths: the future is not determined (as in e.g. dynamical systems with uncertainties, differential inclusions etc.)
- Intuitively: logical operators on paths, and on states

Syntactically

- $A$ means “for all paths” in the future
- $E$ means “exist a path” in the future
- This is combined with $\Box$ and $\Diamond$
- Example (reachability): $E\Diamond p$, some state satisfying $p$ is reachable in some future path
- Example (safety: $p$ is invariant): $A\Box p$, all states in all possible futures satisfy $p$

As used in e.g. UPPAAL

- Henzinger, Sifakis et al. 1992
- Several fragments and syntax, basically, add time constraints
Examples

Differential inclusions

Consider an uncertain dynamical system, e.g. a differential inclusion:

\[ \dot{x} \in F(x) \]
Viability property

For all initial states in $K$, there exists a path such that in the future, $x \in K$ (the viability kernel):

$$x(0) \in K \Rightarrow (E\Box (x \in K))$$
Semantics of TCTL

State formulas

- $\Box_I p$ is true (on state $x$) iff $\forall t \in I, p(t, x)$ is true.
- $\Diamond_I p$ is true (on state $x$) iff $\exists t \in I, p(t, x)$ is true.
- Note that $\neg \Box_I p = \Diamond_I \neg p$.

Path formulas

- $A p$ is true iff $\forall x, p(t, x)$ is true.
- $E p$ is true iff $\exists x$ such that $p(x, t)$ is true.
- Note also $\neg (A p) = E (\neg p)$.

Combining modalities: examples

- $A \Box_I p : \forall x, \forall t \in I, p$ is true.
- $A \Diamond_I p : \forall x, \exists t \in I$ (potentially depending on $x$ then!), $p$ is true.
- $E \Box_I p : \exists x, \forall t \in I, p$ is true ($x$ does not depend on $t$!)
- $E \Diamond_I p : \exists x, \exists t \in I, p$ is true.

For now, we will only discuss these properties.
Abstract model-checking

Abstraction

Now, we only have approximations of the trajectories $g(t, x_0)$, solution of the IVP $\dot{x} = f(x)$, $x(0) = x_0 \in x_0$ : suppose we have on time interval $I$ an outer-approximation function $\overline{g} : I \times \mathcal{P}(\mathbb{R}^n) \rightarrow \mathcal{P}(\mathbb{R}^n)$ and an inner-approximation function $\underline{g} : I \times \mathcal{P}(\mathbb{R}^n) \rightarrow \mathcal{P}(\mathbb{R}^n)$ (on time $\times$ initial condition, generally in box $x_0$), (semi-)decide some of TCTL formulas!

Most notable related work

- Combination of abstract interpretation and model-checking (Clarke, Grumberg & Long 1992) but not applied on numerical properties in general
- Falsification methods on MTL and hybrid systems (Sankaranarayanan & al. S-Taliro tool 2011) but only on one execution
- Monitoring, falsification, parameter synthesis of STL specs. on hybrid systems (Breach, Alexandre Donzé 2010)
- Monitoring of BLTL on hybrid systems (Goldsztejn et al. 2015)
Interpreting TCTL

**Sufficient conditions (proof)**
- If $p$ is true on $\overline{g}(l, x_0)$ then $A□_l p$ is true (we will see later how to do this)
- If $\{x \in \mathbb{R}^n | p(x)\} \cap \overline{g}(l, x_0) \neq \emptyset$ then $E◊_l p$ is true
- If $\exists t \in I$ such that $p$ is true on $\overline{g}(t, x_0)$ then $A◊_l p$
- If $p$ is true on $\overline{g}(l, x_0)$ then $E□_l p$ (strong condition, we can do better, later)

**Necessary conditions (falsification)**
- $E□_l p$ implies $\overline{g}(l, x_0) \cap \{x \in \mathbb{R}^n | p(x)\} \neq \emptyset$
- $A◊_l p$ implies $\overline{g}(l, x_0) \cap \{x \in \mathbb{R}^n | p(x)\} \neq \emptyset$

**Use of negation**
- Use $\neg A□_l p = E◊_l \neg p$; a sufficient condition on the latter (through the inner-approximation) implies that $A□ p$ is false (whereas the interpretation using outer-approximation can only prove $A□ p$ to be true)
- Similarly, we can use $\neg A◊_l p = E□_l \neg p$
Interpreting TCTL

Sufficient conditions

- $A \square_{[5,7]} (x_1 \geq 180)$ ($x \geq 180$ is true on $\bar{g}(l, x_0)$)

$\bar{g}(l, x_0)$
Interpreting TCTL

Sufficient conditions

- \( A\Diamond_I(x_1 \geq 185) \) (\( \exists t \in I \) such that \( p \) is true on \( g(t, x_0) \))
- \( E\Box_I(x_1 \geq 185) \) (\( x_1 \geq 185 \) is true on \( g(I, x_0) \))
- We cannot prove \( A\Box_I(x_1 \geq 185) \) is true, or false
Interpreting TCTL

Sufficient conditions

- \( E \Diamond_I(x_1 \geq 190) \ (\{x_1 \geq 190\} \cap g(I, x_0) \neq \emptyset) \)

Use of negation to falsify \( A \Box_I(x_1 \geq 190) \)

- Use \( \neg A \Box_I(x_1 \geq 190) = E \Diamond_I(x_1 < 190) \)
- \( E \Diamond_I(x_1 < 190) \ (\{x_1 < 190\} \cap g(I, x_0) \neq \emptyset) \) hence \( A \Box_I(x_1 \geq 190) \) is false
Problem statement (ODE)

- For ODE
  \[ \dot{x} = f(x) \text{ with } f : \mathbb{R}^n \to \mathbb{R}^n \]
- Suppose it has a unique solution on time interval \([0, T]\), for an initial condition \(x_0\) at time 0.
- Suppose \(g : \mathbb{R} \times \mathbb{R}^n \to \mathbb{R}^n\) is the \(C^1\) function such that \(t \to g(t, x_0)\) is the solution to this equation with initial condition \(x_0 \in \mathbb{R}^n\).

Outer-approximation

- Use Taylor method (Moore, Berz & Makino etc.) for outer-approximating solution \(g\) to the ODE at some order \(m\):
  \[ g_j(t, x_0) = x_{0,j} + \sum_{i=1}^{m} \frac{f_j^{(i)}(0, x_0)}{i!} t^i + \frac{f_j^{(m+1)}(\xi, x_0)}{(m+1)!} t^{m+1} \]

  where \(f^{(i)}\) is defined inductively as follows:
  \[ f_j^{(l+1)}(t, x_0) = \sum_{i=1}^{n} \frac{\partial f_j^{(l)}}{\partial x_i} f_i(g(t, x_0)) \]
Bounding the remainder: Picard-Lindelöf iteration

### Integral operator

- Let $F(g)(t, x) = x_0 + \int_0^t f(g(s, x)) \, ds$
- Under simple hypotheses, the fixed point of $F$ (on a small time interval $[0, T]$) exists and can be computed by iteration of $F$, and is the solution to our ODE

### Rough enclosures

- Can also be used also to estimate the remainder $\frac{f^{(m+1)}(\xi, x_0)}{(m+1)!} (\xi \in [0, T])$ since this depends on $g(\xi, x_0)$
- Suppose we have $x$, an interval such that $x_0 + [0, T]f(x) \subseteq x$

then $[0, T] \times x$ contains all points $g(t, x_0)$, where $g$ is a solution to our ODE on $[0, T]$, $t \in [0, T]$ and $x_0 \in x_0$. 
For inner-approximations

- To compute inner-approximation, we need outer-approximations of the Jacobian of the solution, with respect to initial conditions, as in the discrete case.

- The variational equation (as in e.g. Wilczak) is the ODE that is satisfied by \( g \) and its Jacobian \( J_{j,i}^g = \frac{\partial g_j}{\partial x_{0,i}} \) with respect to the initial condition \( x_0 \):

\[
\begin{align*}
\frac{dg}{dt}(t, x_0) &= f(g(t, x_0)) \\
\frac{dJ_{j,i}^g}{dt}(t, x_0) &= \sum_{k=1}^{n} \frac{\partial f_j}{\partial x_k}(g(t, x_0)) J_{k,i}^g(t, x_0)
\end{align*}
\]

Furthermore, the initial condition that \( g_j \) and \( J_{j,i}^g \) satisfy are:

\[
\begin{align*}
 g_j(0, x_0) &= x_{0,j} \\
 J_{j,i}^g(0, x_0) &= \delta_{i,j}
\end{align*}
\]

where \( \delta_{i,j} \) is the Kronecker symbol.
A simple ODE with uncertain initial values

- Consider the ODE \( \dot{x} = x \) with \( x_0 \in [0, 1] \) and \( t \in [0, \frac{1}{2}] \).
- The variational equation associated to this ODE is (noting \( J^g = J^g_{1,1} \)) :

\[
\frac{dg}{dt}(t, x_0) = g(t, x_0) \\
\frac{dJ^g}{dt}(t, x_0) = J^g(t, x_0)
\]

Furthermore, the initial condition that \( g \) and \( J^g \) satisfy are :

\[
g(0, x_0) = x_0 \\
J^g(0, x_0) = 1
\]

Rough enclosures

- We see that \( x = [0, 2] \) satisfies \( [0, 1] + [0, \frac{1}{2}] [0, 2] \subseteq [0, 2] \) hence for all \( t \in [0, \frac{1}{2}] \), for all \( x_0 \in [0, 1] \), \( g(t, x_0) \in [0, 2] \).
- Furthermore, we see that \( x = [1, 2] \) satisfies \( 1 + [0, \frac{1}{2}] [1, 2] \subseteq [1, 2] \) so for all \( t \in [0, \frac{1}{2}] \), for all \( x_0 \in [0, 1] \), \( J^g(t, x_0) \in [1, 2] \).
Outer-approximation for the solutions (order 3)

Taylor model

\[ g(t, x_0) = x_0 + x_0 t + \frac{x_0}{2} t^2 + \frac{g(\xi, x_0)}{6} t^3 \]

on \( x_0 \in x_0 = \frac{1}{2} + \frac{1}{2} \varepsilon_1 \in [0, 1] \):

\[ \overline{g}(t, x_0) = \left( \frac{1}{2} + \frac{1}{2} \varepsilon_1 \right) \left( 1 + t + \frac{t^2}{2} \right) + [0, \frac{2}{3}] t^3 \]

For instance, at time \( \frac{1}{2} \)

\[ \overline{g} \left( \frac{1}{2}, x_0 \right) = \left[ \frac{13}{16}, \frac{41}{48} \right] + \frac{13}{16} \varepsilon_1 \]

Hence \([0, e^{\frac{1}{2}}] \sim [0, 1.64872] \subseteq \overline{g} \left( \frac{1}{2}, x_0 \right) = \left[ 0, \frac{5}{3} \right] \sim [0, 1.66667] \)
Outer-approximation of the Jacobian (order 3), appli to inner-approx

Taylor model for the Jacobian

- We can outer-approximate the Jacobian, for all \( t \in [0, \frac{1}{2}] \):
  
  \[
  J^g(t, x_0) = 1 + t + \frac{t^2}{2} + \frac{J^g(\xi, x_0)}{6} t^3
  \]
  \[
  \overline{J^g}(t, x_0) = 1 + t + \frac{t^2}{2} + \frac{[1, 2]}{6} t^3
  \]

Example: inner-approximation at time \( t = \frac{1}{2} \)

- \( \overline{J^g} \left( \frac{1}{2}, x_0 \right) \subseteq \left[ \frac{79}{48}, \frac{5}{3} \right] \)
- Mean value theorem, evaluated in \( \varepsilon_1 = 0 \), that is \( x_0 = \text{mid}(x_0) = \frac{1}{2} \), at time \( t = \frac{1}{2} \), yields an inner-approximation of \( \{ g(\frac{1}{2}, x_0), x_0 \in x_0 = [0, 1] \} \):
  
  \[
  g \left( \frac{1}{2}, x_0 \right) = \text{pro} \left( \overline{g} \left( \frac{1}{2}, \frac{1}{2} \right) + \frac{1}{2} \overline{J^g} \left( \frac{1}{2}, x_0 \right) [1, -1] \right)
  \]
  
  \[
  = \text{pro} \left( \left[ \begin{array}{c}
  \frac{13}{16}, \frac{41}{48} \\
  \end{array} \right] + \frac{1}{2} \left[ \begin{array}{c}
  \frac{79}{48}, -\frac{79}{48} \\
  \end{array} \right] \right) \]
  
  \[
  = \text{pro} \left( \left[ \begin{array}{c}
  \frac{157}{96}, \frac{3}{96} \\
  \end{array} \right] \right)
  \]
  
  \[
  \left[ \begin{array}{c}
  \frac{3}{96}, \frac{157}{96} \\
  \end{array} \right] \sim [0.03, 1.635]
  \]
  
  \[
  \subseteq \{ g \left( \frac{1}{2}, x_0 \right), x_0 \in x_0 \} = \left[ 0, e^{\frac{1}{2}} \right] \sim [0, 1.649]
  \]
Quality of the approximations: example of the Brusselator

\[
\begin{align*}
\dot{x}_1 &= 1 + x_1^2 x_2 - 2.5x_1 \\
\dot{x}_2 &= 1.5x_1 - x_1^2 x_2
\end{align*}
\]

with \( x_1(0) \in [0.9, 1] \) and \( x_2(0) \in [0, 0.1] \).
When the result of adding the (proper) over-approx of $g(x_0,t)$ to the (improper) $\text{Jac}(X_0,t)^*[1,-1]$ is proper, then the inner-approximation is empty.
For this, we have to compute:

\[ \bar{g}(t, \text{mid}(x_0)) + \frac{1}{2} \bar{J}g(t, x_0)[1, -1] \subseteq \{g(t, x_0), x_0 \in x_0 = [0, 1]\} \]

So we need the center, for all \( t \):

First,

\[ \bar{g}\left(t, \frac{1}{2}\right) = \left[\frac{1}{2} + \frac{1}{2}t + \frac{1}{4}t^2, \frac{1}{2} + \frac{1}{2}t + \frac{1}{4}t^2 + \frac{1}{3}t^3\right] \]

And the outer-approximation of the Jacobian, for all \( t \) and \( x_0 \in x_0 = [0, 1] \):

\[ \bar{J}g(t, x_0) \subseteq \left[1 + t + \frac{t^2}{2} + \frac{1}{6}t^3, 1 + t + \frac{t^2}{2} + \frac{1}{3}t^3\right] \]

Therefore

For all \( t \in [0, \frac{1}{2}] \),

\[ g(t, [0, 1]) = \left[\frac{t^3}{4}, 1 + t + \frac{t^2}{2} + \frac{t^3}{12}\right] \subseteq \{g(t, x_0), x_0 \in [0, 1]\} \]
Function $g(t, x)$ and application

Consider $E\left[\frac{1}{4}, \frac{1}{2}\right] (y > 1.5)$.

It will be true if $g\left(\left[\frac{1}{4}, \frac{1}{2}\right], [0, 1]\right)$ intersects $y > 1.5$.

We see that

$$g\left(\left[\frac{1}{4}, \frac{1}{2}\right], [0, 1]\right) = \left[\frac{1}{64}, 1.5 + \frac{7}{48}\right] \cap [1.5, \infty] \neq \emptyset$$
In general

Algorithmically

- We produce Taylor models for solutions and Jacobians (between all \([kT, (k + 1)T]\))
- We deduce on all these time intervals \(\bar{g}\) and \(g\) which are polynomial in \(t\) with coefficients in affine forms (linking them to uncertain initial values and parameters)
- For predicates \(p = (f(t, x) \geq 0)\) with \(f\) polynomial, deciding \(p\) is true on \(g(I, x_0)\) (resp. \(\bar{g}(I, x_0)\)) can be done by any interval method (direct evaluation, affine forms, Bernstein polynomials etc.)
- For such predicates, deciding non-emptiness of the intersection of \(\{x \in \mathbb{R}^n | p(x)\}\) with \(g(I, x_0)\) (resp. \(\bar{g}(I, x_0)\)) is a constraint satisfaction problem

Symbolic expressions in \(t\) and the \(\epsilon_i\) allow refinements: example of \(E\Box_I p\)

- A sufficient condition is \(p\) true on \(g(I, x_0)\)
- If \(p = (f(t, x) \geq 0)\), a finer criterion is to check the non-emptiness of the constraint on \(\epsilon_i : f(I, g(I, x_0)) \geq 0\) (polynomial in \(\epsilon_i\)
Example: aircraft (Lygeros 2011 & Sankaranarayanan 2014)

Differential model

\[
\begin{align*}
\dot{x}_1 &= -\frac{S\rho B_0}{2m} x_1^2 - g \sin\left(\frac{\pi x_1}{180}\right) + \frac{u_1}{m} - \frac{S\rho}{2m} x_1^2 (B_1 u_2 + B_2 u_2^2) \\
\dot{x}_2 &= \frac{S\rho C_0}{2m} x_1 - g \cos\left(\frac{\pi x_1}{180}\right) + \frac{S\rho C_1}{2m} x_1 u_2 \\
\dot{x}_3 &= x_1 \sin\left(\frac{\pi x_2}{180}\right)
\end{align*}
\]

with initial conditions \(x_1(0) \in [200, 260]\), \(x_2(0) \in [-10, 10]\), \(x_3(0) \in [120, 150]\).

- \(x_1\): speed, \(x_2\): angle, \(x_3\): altitude
- the inputs \(u_1 \in [0.1, 0.2]\) and \(u_2 \in [0.1, 0.2]\) represent respectively the thrust and the angle of attack.
- Constants: \(B_0 = 0.07351\), \(B_1 = -0.0015\), \(B_2 = 0.00061\), \(C_0 = 0.1667\), \(C_1 = 0.109\), \(m = 74000\), \(g = 9.81\), \(S = 158\), \(\rho = 0.3804\)
- The model is correct for small angle approximation for \(u_2\).

Typical temporal properties to be checked (Sankaranarayanan 2013)

\[\neg (\square_{[0.5,1.5]} a \land \diamond_{[3,4]} b), \neg (\square_{[0,4]} a \land \diamond_{[3.5,4]} d), \neg \diamond_{[1,3]} e), \neg (\diamond_{[0.5,1]} a \land \square_{[3,4]} g), \neg (\square_{[0,5]} h), \neg (\square_{[2,2.5]} (i_1 \land i_2))\] where \(a\) is \(240 \leq x_1 \leq 250\), \(b\) is \(230 \leq x_1 \leq 240\), \(d\) is \(240 \leq x_1 \leq 240.1\), \(e\) is \(x_1 \geq 260\), \(g\) is \(270 \leq x_1 \leq 280\), \(h\) is \(190 \leq x_1 \leq 210\), \(i_1\) is \(190 \leq x_1 \leq 200\), \(i_3\) is \(190 \leq x_3 \leq 200\).
Inner-approximation: there exist trajectories for which \( (\Box_{[0,1]} a \land \Diamond_{[5,7]} b) \) is true.

\[ E(\Box_{[0,1]} 200 \leq x_1 \leq 210 \land \Diamond_{[5,7]} x_1 \leq 190) \]
Example: aircraft (Lygeros 2011 & Sankaranarayanan 2014)

Does there also exist trajectories for which \((\square_{[0.1]} a \land \Diamond_{[5,7]} b)\) is false?

\[
E(\neg(\Diamond_{[5,7]} x_1 \leq 190))?
\]

\[
E(\square_{[5,7]} x_1 > 190)???
\]

Can I conclude?

\[
E(\square_{[0,1]} 200 \leq x_1 \leq 210 \land \Diamond_{[5,7]} x_1 \leq 190)
\]
Example: aircraft (Lygeros 2011 & Sankaranarayanan 2014)

Does there also exist trajectories for which \((\square_{0..1.0} a \land \Diamond_{[5,7]} b)\) is false?

\[
E(\square_{[5,7]} x_1 > 190) \text{?}
\]

I do not know!

=> refinement of initial state
Future work

Extension to the full logics

- Interpret the full fragment $A/E(stateformula)$ of TCTL
- Will need propagation of time constraints in the vein of (Ishii, Yonezaki, Goldsztejn 2015 - BLTL)

Any questions?