AMT: a Property-based Tool for Monitoring Analog Systems

Dejan Ničković
Verimag

Oded Maler
Verimag

Overview

- Introduction
- STL/PSL Specification Language
  - Analog Layer
  - Temporal Layer
  - Distance-based Operators
- Checking STL/PSL Properties
  - Offline Marking
  - Incremental Marking
- AMT Tool
- FLASH Memory Case Study

Introduction

- Verification of discrete systems
  - Model checking TL specs
  - Central role in algorithmic verification
  - Efficient algorithms for LTL, CTL, PSL etc.
- Verification of real-time systems
  - Emptiness checking of *timed automata*
    - KRONOS, UPPAAL, IF etc.
  - Many variants of real-time logics
    - MTL, MITL, TCTL etc.
  - Only TCTL used in a real-time verification tool
- Lightweight verification
  - Systems may be too complex to verify exhaustivly
    - Software
    - Very large digital systems
    - Many real-time systems etc.
  - Property monitors
    - Generated automatically from the specification
    - Observe individual simulation traces and check whether the property is violated
    - Incomplete but more reliable method than manual visual inspection of simulation traces

Introduction

- Verification of discrete systems
  - Model checking TL specs
  - Central role in algorithmic verification
  - Efficient algorithms for LTL, CTL, PSL etc.
- Verification of real-time systems
  - Emptiness checking of *timed automata*
    - KRONOS, UPPAAL, IF etc.
  - Many variants of real-time logics
    - MTL, MITL, TCTL etc.
  - Only TCTL used in a real-time verification tool

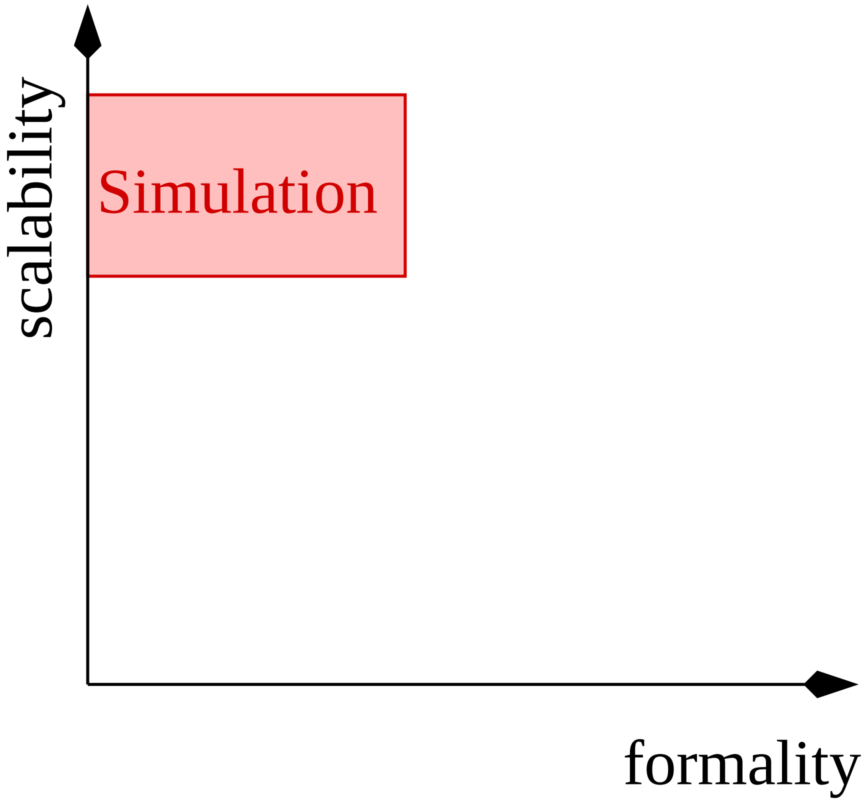
- Lightweight verification
  - Systems may be too complex to verify exhaustivly
    - Software
    - Very large digital systems
    - Many real-time systems etc.
  - Property monitors
    - Generated automatically from the specification
    - Observe individual simulation traces and check whether the property is violated
    - Incomplete but more reliable method than manual visual inspection of simulation traces

Introduction

- Verification of discrete systems
  - Model checking TL specs
  - Central role in algorithmic verification
  - Efficient algorithms for LTL, CTL, PSL etc.
- Verification of real-time systems
  - Emptiness checking of *timed automata*
    - KRONOS, UPPAAL, IF etc.
  - Many variants of real-time logics
    - MTL, MITL, TCTL etc.
  - Only TCTL used in a real-time verification tool
- Lightweight verification
  - Systems may be too complex to verify exhaustivly
    - Software
    - Very large digital systems
    - Many real-time systems etc.
  - Property monitors
    - Generated automatically from the specification
    - Observe individual simulation traces and check whether the property is violated
    - Incomplete but more reliable method than manual visual inspection of simulation traces

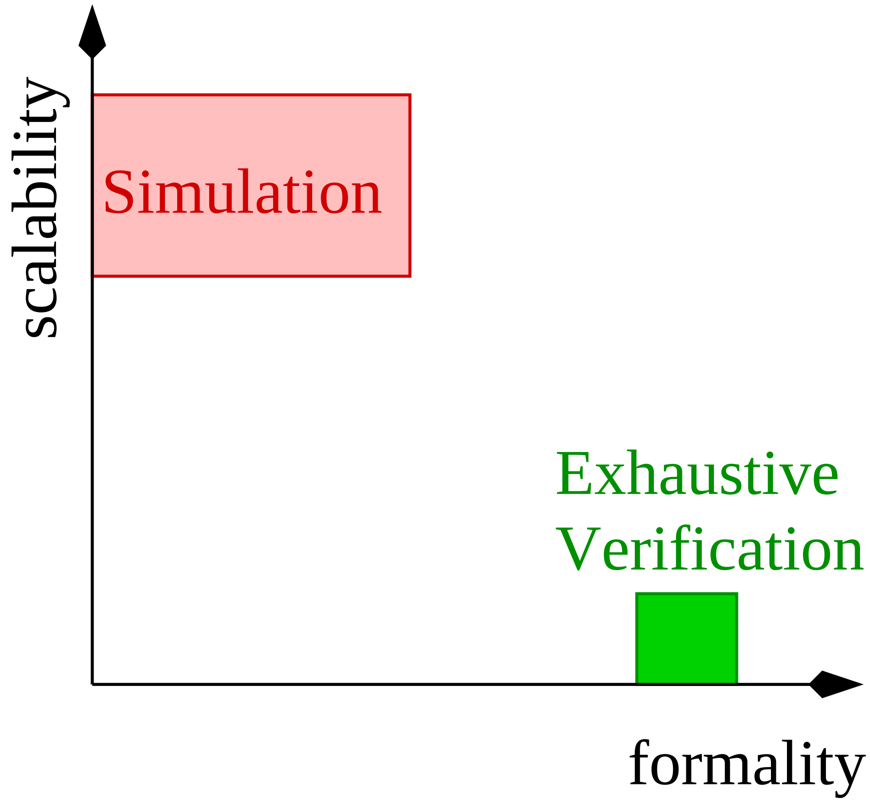
Introduction

- Verification of continuous systems
  - Manual inspection of simulation traces
    - Dominant technique
    - Requires experienced specialists
    - Error prone
  - Exhaustive analog verificaiton
    - Powerful formalisms such as hybrid automata
    - Limited scalability
- **Our approach:** Property-based lightweigh verification of continuous signals



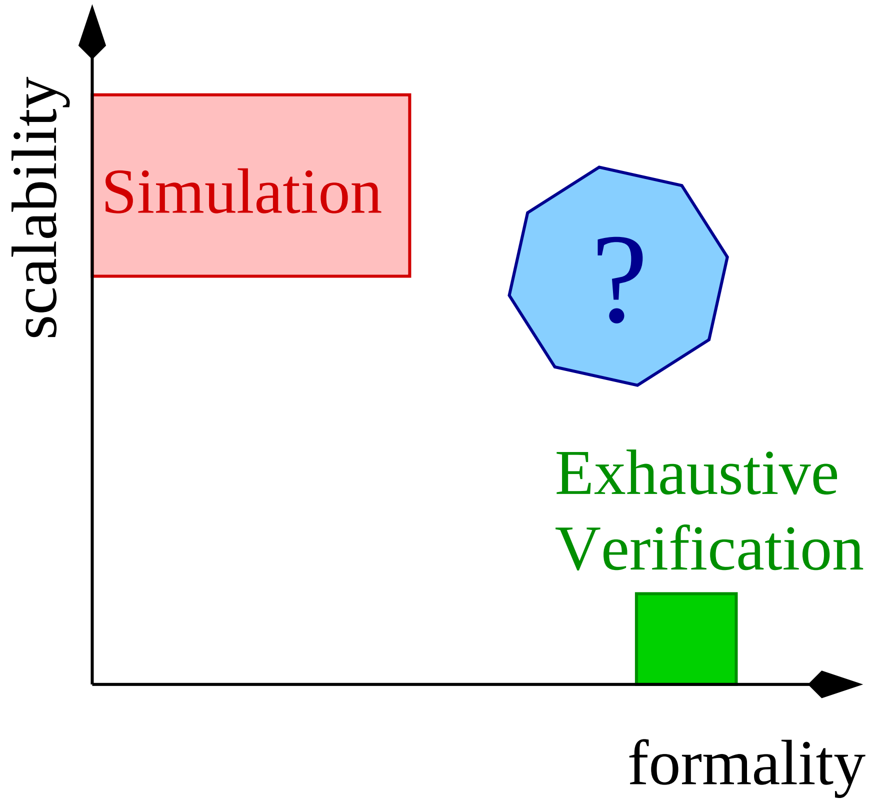
Introduction

- Verification of continuous systems
  - Manual inspection of simulation traces
    - Dominant technique
    - Requires experienced specialists
    - Error prone
  - Exhaustive analog verificaiton
    - Powerful formalisms such as hybrid automata
    - Limited scalability
- **Our approach:** Property-based lightweigh verification of continuous signals



Introduction

- Verification of continuous systems
  - Manual inspection of simulation traces
    - Dominant technique
    - Requires experienced specialists
    - Error prone
  - Exhaustive analog verificaiton
    - Powerful formalisms such as hybrid automata
    - Limited scalability
- **Our approach:** Property-based lightweigh verification of continuous signals



Signals

- Finite length signal ξ defined over an abstract domain $\mathbb{D}$
  - Partial function $\xi : \mathbb{T} \rightarrow \mathbb{D}$
  - Length of ξ is r ($|\xi| = r$)
  - $\xi[t] = \perp$ when $t \geq |\xi|$
  - **Boolean signals:** (ξ_b) $\mathbb{D} = \mathbb{B}$
  - **Continuous signals:** (ξ_a) $\mathbb{D} = \mathbb{R}$
- **Restriction** of a signal ξ to length d

$$\xi' = \langle \xi \rangle_d \text{ iff } \xi'[t] = \begin{cases} \xi[t] & \text{if } t < d \\ \perp & \text{otherwise} \end{cases}$$

- **Concatenation** $\xi = \xi_1 \cdot \xi_2$

$$\xi[t] = \begin{cases} \xi_1[t] & \text{if } t < r_1 \\ \xi_2[t - r_1] & \text{otherwise} \end{cases}$$

- d**-suffix** of a signal ξ, $\xi' = d \backslash \xi$

$$\xi'[t] = \xi[t + d] \quad \text{for every } t \in [0, |\xi| - d)$$

Signals

- **Minkowski sum** and **difference** of two sets P_1 and P_2 are defined as

$$P_1 \oplus P_2 = \{x_1 + x_2 : x_1 \in P_1, x_2 \in P_2\}$$
$$P_1 \ominus P_2 = \{x_1 - x_2 : x_1 \in P_1, x_2 \in P_2\}.$$

- **Projection** of the signal ξ on the dimension with domain $\mathbb{B}$ which corresponds to the proposition p, $\xi_p = \pi_p(\xi)$
  - Likewise $\xi_s = \pi_s(\xi)$ is the projection of the signal ξ on the dimension with domain $\mathbb{R}$ which corresponds to the continuous variable s
- **Signal representation**
  - Boolean signals:
    - Non-Zeno finite length signals admit finite representation
    - Sequence of adjacent intervals with value constant in each interval
  - Continuous signals:
    - Do not admit an exact finite representation
    - But, numerical simulators produce a **finite** collection of sampling points
    - The signal value at missing points in time is interpolated

STL/PSL *Specification Language*

- Extension of real-time temporal logic MITL with analog constructs
- PSL-like layered approach
  - **Analog layer:** allows reasoning about continuous signals
  - **Temporal layer:** relates the temporal behavior of input traces
- “Communication” between two layers via **static abstractions**
  - Partitioning of the continuous state space according to the satisfaction of some inequality constraints on the continuous variables
- Targeted to be used in *lightweight* verification
  - PSL-like **finitary interpratation** of temporal operators

STL/PSL *Specification Language*

- Extension of real-time temporal logic MITL with analog constructs
- PSL-like layered approach
  - **Analog layer:** allows reasoning about continuous signals
  - **Temporal layer:** relates the temporal behavior of input traces
- "Communication" between two layers via **static abstractions**
  - Partitioning of the continuous state space according to the satisfaction of some inequality constraints on the continuous variables
- Targeted to be used in *lightweight* verification
  - PSL-like **finitary interpratation** of temporal operators

STL/PSL *Specification Language*

- Extension of real-time temporal logic MITL with analog constructs
- PSL-like layered approach
  - **Analog layer:** allows reasoning about continuous signals
  - **Temporal layer:** relates the temporal behavior of input traces
- "Communication" between two layers via **static abstractions**
  - Partitioning of the continuous state space according to the satisfaction of some inequality constraints on the continuous variables
- Targeted to be used in *lightweight* verification
  - PSL-like **finitary interpratation** of temporal operators

STL/PSL *Specification Language*

- Extension of real-time temporal logic MITL with analog constructs
- PSL-like layered approach
  - **Analog layer:** allows reasoning about continuous signals
  - **Temporal layer:** relates the temporal behavior of input traces
- "Communication" between two layers via **static abstractions**
  - Partitioning of the continuous state space according to the satisfaction of some inequality constraints on the continuous variables
- Targeted to be used in *lightweight* verification
  - PSL-like **finitary interpratation** of temporal operators

- **Syntax:**

$$\phi :== s \mid \texttt{shift}(\phi, \mathsf{k}) \mid \phi_1 \star \phi_2 \mid \phi \star \mathsf{c} \mid \texttt{abs}(\phi)$$

where s belongs to a set $S = \{s_1, s_2, \ldots, s_n\}$ of continuous variables, $\star \in \{\texttt{+}, \texttt{-}, \texttt{*}\}$, $\mathsf{c} \in \mathbb{Q}$ and $\mathsf{k} \in \mathbb{Q}^+$.

- **Semantics:**

$$\begin{array}{lcl} s[t] & = & \pi_s(\xi)[t] \\ \texttt{shift}(\phi, \mathsf{k})[t] & = & \phi[t + \mathsf{k}] \\ (\phi_1 \star \phi_2)[t] & = & \phi_1[t] \star \phi_2[t] \\ (\phi \star \mathsf{c})[t] & = & \phi[t] \star \mathsf{c} \\ \texttt{abs}(\varphi)[t] & = & \begin{cases} \phi[t] & \text{if } \phi[t] \geq 0 \\ -\phi[t] & \text{otherwise} \end{cases} \end{array}$$

- Pragmatic choice of analog operators
  - Based on the feedback of analog designers
  - Can be naturally extended

STL/PSL: *Analog Layer*

- **Syntax:**

$$\phi :== s \mid \texttt{shift}(\phi, \texttt{k}) \mid \phi_1 \star \phi_2 \mid \phi \star \texttt{c} \mid \texttt{abs}(\phi)$$

where s belongs to a set $S = \{s_1, s_2, \ldots, s_n\}$ of continuous variables, $\star \in \{\texttt{+}, \texttt{-}, \texttt{*}\}$, c $\in \mathbb{Q}$ and k $\in \mathbb{Q}^+$.

- **Semantics:**

$$\begin{array}{lcl} s[t] & = & \pi_s(\xi)[t] \\ \texttt{shift}(\phi, \texttt{k})[t] & = & \phi[t + \texttt{k}] \\ (\phi_1 \star \phi_2)[t] & = & \phi_1[t] \star \phi_2[t] \\ (\phi \star \texttt{c})[t] & = & \phi[t] \star \texttt{c} \\ \texttt{abs}(\varphi)[t] & = & \begin{cases} \phi[t] & \text{if } \phi[t] \geq 0 \\ -\phi[t] & \text{otherwise} \end{cases} \end{array}$$

- Pragmatic choice of analog operators
  - Based on the feedback of analog designers
  - Can be naturally extended

- **Syntax:**

$$\phi :== s \mid \texttt{shift}(\phi, \texttt{k}) \mid \phi_1 \star \phi_2 \mid \phi \star \texttt{c} \mid \texttt{abs}(\phi)$$

where s belongs to a set $S = \{s_1, s_2, \ldots, s_n\}$ of continuous variables, $\star \in \{\texttt{+}, \texttt{-}, \texttt{*}\}$, c $\in \mathbb{Q}$ and k $\in \mathbb{Q}^+$.

- **Semantics:**

$$\begin{array}{lcl} s[t] & = & \pi_s(\xi)[t] \\ \texttt{shift}(\phi, \texttt{k})[t] & = & \phi[t + \texttt{k}] \\ (\phi_1 \star \phi_2)[t] & = & \phi_1[t] \star \phi_2[t] \\ (\phi \star \texttt{c})[t] & = & \phi[t] \star \texttt{c} \\ \texttt{abs}(\varphi)[t] & = & \begin{cases} \phi[t] & \text{if } \phi[t] \geq 0 \\ -\phi[t] & \text{otherwise} \end{cases} \end{array}$$

- Pragmatic choice of analog operators
  - Based on the feedback of analog designers
  - Can be naturally extended

- **Syntax:**

$$\varphi :== p \mid \phi \circ \texttt{c} \mid \texttt{not}\ \varphi \mid \varphi_1\ \texttt{or}\ \varphi_2 \mid \texttt{eventually!}\ \varphi \mid \texttt{eventually![a:b]}\ \varphi \mid \texttt{eventually[a:b]}\ \varphi \mid \varphi_1\ \texttt{until!}\ \varphi_2 \mid \varphi_1\ \texttt{until![a:b]}\ \varphi_2$$

where p belongs to a set $P = \{p_1, p_2, \ldots, p_n\}$ of propositional variables, $\texttt{a}, \texttt{b}, \texttt{c} \in \mathcal{Q}$ and $\circ \in \{\texttt{>}, \texttt{>=}, \texttt{<}, \texttt{<=}\}$.

- **Semantics:** The satisfaction relation $(\xi, t) \models \varphi$, indicating that signal ξ satisfies φ at time t is defined inductively as follows:

$$
\begin{array}{lll}
(\xi,t) \models \texttt{eventually!}\ \varphi & \text{iff} & \exists t' \geq t \ \texttt{st}\ t' < |\xi| \ \texttt{and}\ (\xi,t') \models \varphi \\
(\xi,t) \models \texttt{eventually![a:b]}\ \varphi & \text{iff} & \exists t' \in t \oplus [a,b] \ \texttt{st}\ t' < |\xi| \ \texttt{and}\ (\xi,t') \models \varphi \\
(\xi,t) \models \varphi_1\ \texttt{until!}\ \varphi_2 & \text{iff} & \exists t' \geq t \text{ st } t' < |\xi| \text{ and } (\xi,t') \models \varphi_2 \text{ and} \\
 & & \forall t'' \in [t,t']\ (\xi,t'') \models \varphi_1 \\
(\xi,t) \models \varphi_1\ \texttt{until![a:b]}\ \varphi_2 & \text{iff} & \exists t' \in t \oplus [a,b] \text{ st } t' < |\xi| \text{ and } (\xi,t') \models \varphi_2 \text{ and} \\
 & & \forall t'' \in [t,t']\ (\xi,t'') \models \varphi_1
\end{array}
$$

- **Syntax:**

$$\varphi ::= p \mid \phi \circ \texttt{c} \mid \texttt{not}\ \varphi \mid \varphi_1\ \texttt{or}\ \varphi_2 \mid \texttt{eventually!}\ \varphi \mid \texttt{eventually![a:b]}\ \varphi \mid \texttt{eventually[a:b]}\ \varphi \mid \varphi_1\ \texttt{until!}\ \varphi_2 \mid \varphi_1\ \texttt{until![a:b]}\ \varphi_2$$

  where p belongs to a set $P = \{p_1, p_2, \ldots, p_n\}$ of propositional variables, $\texttt{a}, \texttt{b}, \texttt{c} \in \mathcal{Q}$ and $\circ \in \{\texttt{>}, \texttt{>=}, \texttt{<}, \texttt{<=}\}$.
- **Semantics:** The satisfaction relation $(\xi, t) \models \varphi$, indicating that signal ξ satisfies φ at time t is defined inductively as follows:

$$\begin{array}{lll}
(\xi,t) \models \texttt{eventually!}\ \varphi & \text{iff} & \exists t' \geq t\ \texttt{st}\ t' < |\xi|\ \texttt{and}\ (\xi,t') \models \varphi \\
(\xi,t) \models \texttt{eventually![a:b]}\ \varphi & \text{iff} & \exists t' \in t \oplus [a,b]\ \texttt{st}\ t' < |\xi|\ \texttt{and}\ (\xi,t') \models \varphi \\
(\xi,t) \models \varphi_1\ \texttt{until!}\ \varphi_2 & \text{iff} & \exists t' \geq t \text{ st } t' < |\xi| \text{ and } (\xi,t') \models \varphi_2 \text{ and} \\
 & & \forall t'' \in [t,t']\ (\xi,t'') \models \varphi_1 \\
(\xi,t) \models \varphi_1\ \texttt{until![a:b]}\ \varphi_2 & \text{iff} & \exists t' \in t \oplus [a,b] \text{ st } t' < |\xi| \text{ and } (\xi,t') \models \varphi_2 \text{ and} \\
 & & \forall t'' \in [t,t']\ (\xi,t'') \models \varphi_1
\end{array}$$

STL/PSL: *Distance-based Operators*

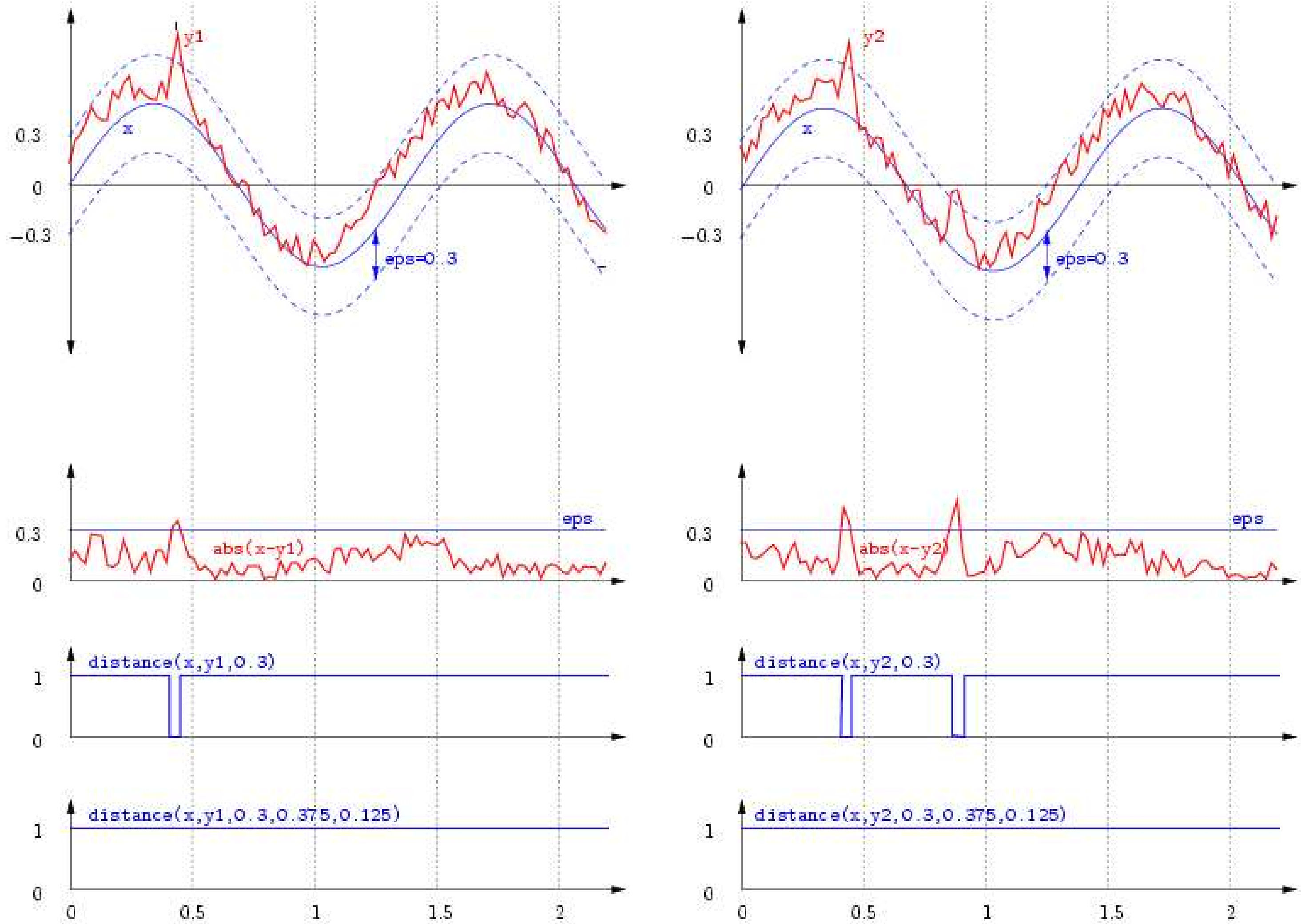
- **Motivation:** Enrich STL/PSL with **metric properties**
- Compare waveforms with some reference signal that specifies a desired behavior
- Distance function (metric)
  - Quantifies numerically the resemblance of two signals

$$\texttt{distance}(\phi_1, \phi_2, \texttt{c}) = \texttt{abs}(\phi_1 \texttt{-} \phi_2) \texttt{ <= c}$$

$$\texttt{distance}(\phi_1, \phi_2, \texttt{c}, \texttt{t}, \texttt{T}) = \texttt{abs}(\phi_1 \texttt{-} \phi_2) \texttt{ > c -> eventually![<=t] always[<=T-t]}(\texttt{abs }(\phi_1 \texttt{-} \phi_2) \texttt{ <= c})$$

$$\texttt{distance}(\varphi_1, \varphi_2, \texttt{t}, \texttt{T}) = (\varphi_1 \texttt{ xor } \varphi_2)\texttt{-> eventually![<=t] always[<=T-t] }(\varphi_1 \texttt{ iff } \varphi_2)$$

STL/PSL: *Distance-based Operators Example*



Checking STL/PSL *Properties*

- **Marking:** a procedure that determines the truth values of each subformula of an STL/PSL specification at every time instant t
  - Doubly-recursive procedure, on time and the structure of the formula
- Two algorithms for checking STL/PSL properties:
  - **Offline marking:** input is fully available
  - **Incremental marking:** input is dynamically observed
- Based on [MalerN04]

Checking STL/PSL *Properties*

- **Marking:** a procedure that determines the truth values of each subformula of an STL/PSL specification at every time instant t
  - Doubly-recursive procedure, on time and the structure of the formula
- Two algorithms for checking STL/PSL properties:
  - **Offline marking:** input is fully available
  - **Incremental marking:** input is dynamically observed
- Based on [MalerN04]

Checking STL/PSL *Properties*

- **Marking:** a procedure that determines the truth values of each subformula of an STL/PSL specification at every time instant t
  - Doubly-recursive procedure, on time and the structure of the formula
- Two algorithms for checking STL/PSL properties:
  - **Offline marking:** input is fully available
  - **Incremental marking:** input is dynamically observed
- Based on [MalerN04]

Offline Marking

input : STL/PSL Temporal Formula φ and signal ξ

switch φ **do**
  case p
    $\chi_\varphi := \pi_p(\xi)$;
  end
  case $\text{OP}_2(\varphi_1, \varphi_2)$
    OFFLINE (φ_1, φ_2);
    $\chi_\varphi :=$ COMBINE$(\text{OP}_2, \chi_{\varphi_1}, \chi_{\varphi_2})$;
  end
end

- Inputs:
  - Multidimentional signal ξ
  - STL/PSL specification φ
- Compute, from **bottom-up**, a signal $\chi_\psi(\xi)$ for each subformila ψ of ϕ
- COMBINE computes from input signals a new signal based on the specific operation

Offline Marking

input : STL/PSL Temporal Formula φ and signal ξ

switch φ **do**
  case p
    $\chi_\varphi := \pi_p(\xi)$;
  end
  case $\text{OP}_2(\varphi_1, \varphi_2)$
    OFFLINE (φ_1, φ_2);
    $\chi_\varphi :=$ COMBINE$(\text{OP}_2, \chi_{\varphi_1}, \chi_{\varphi_2})$;
  end
end

- Inputs:
  - Multidimentional signal ξ
  - STL/PSL specification φ
- Compute, from **bottom-up**, a signal $\chi_\psi(\xi)$ for each subformila ψ of ϕ
- COMBINE computes from input signals a new signal based on the specific operation

Offline Marking

---

input : STL/PSL Temporal Formula φ and signal ξ

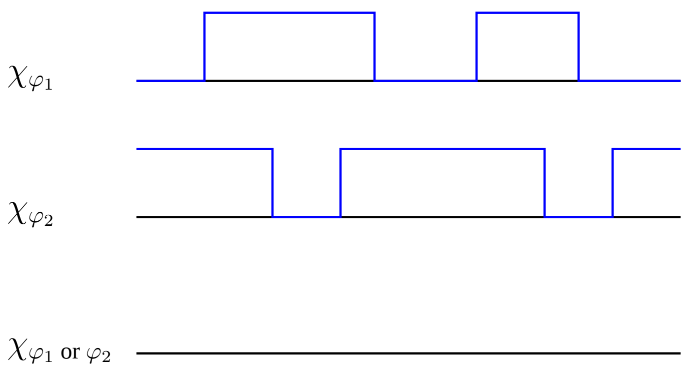
switch φ **do**
  case p
    $\chi_\varphi := \pi_p(\xi)$;
  end
  case $\text{OP}_2(\varphi_1, \varphi_2)$
    OFFLINE (φ_1, φ_2);
    $\chi_\varphi :=$ COMBINE$(\text{OP}_2, \chi_{\varphi_1}, \chi_{\varphi_2})$);
  end
end

- Inputs:
  - Multidimentional signal ξ
  - STL/PSL specification φ
- Compute, from **bottom-up**, a signal $\chi_\psi(\xi)$ for each subformila ψ of ϕ
- COMBINE computes from input signals a new signal based on the specific operation

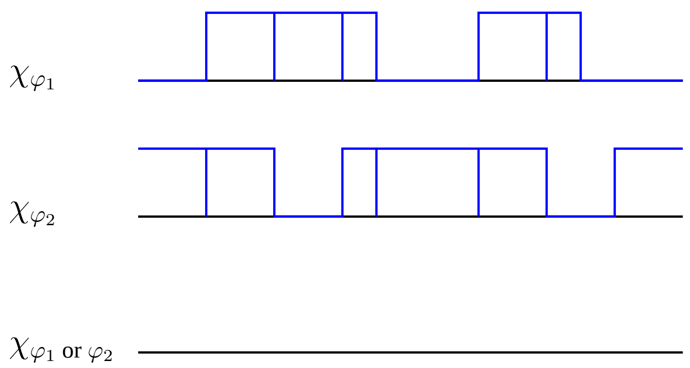
COMBINE: ***Disjunction and Eventually***

- Disjunction
  - Refine the intervals of χ_{ϕ_1} and χ_{ϕ_2} so that the mutual values of both signals become uniform in every interval
  - Compute the disjunction interval-wise
  - Merge the adjacent intervals having the same value

- Bounded Eventually
  - For every positive interval $I \in \chi_{\varphi_1}$
  - Compute its **back shifting** $I - [a,b] \cap \mathbb{T}$
  - Merge the overlapping intervals in χ_{φ}



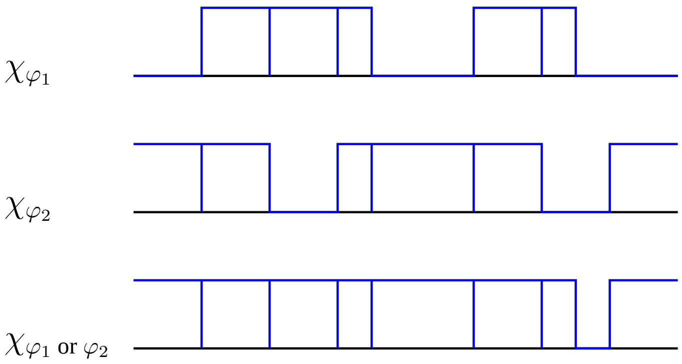
- Disjunction
  - Refine the intervals of χ_{ϕ_1} and χ_{ϕ_2} so that the mutual values of both signals become uniform in every interval
  - Compute the disjunction interval-wise
  - Merge the adjacent intervals having the same value



- Bounded Eventually
  - For every positive interval $I \in \chi_{\varphi_1}$
  - Compute its **back shifting** $I - [a, b] \cap \mathbb{T}$
  - Merge the overlapping intervals in χ_{φ}

Combine: *Disjunction and Eventually*

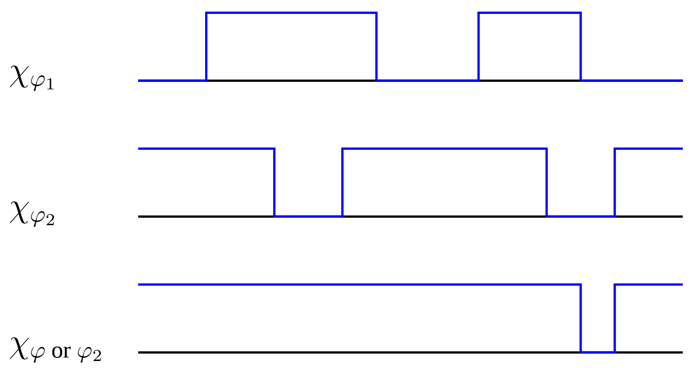
- Disjunction
  - Refine the intervals of χ_{ϕ_1} and χ_{ϕ_2} so that the mutual values of both signals become uniform in every interval
  - Compute the disjunction interval-wise
  - Merge the adjacent intervals having the same value



- Bounded Eventually
  - For every positive interval $I \in \chi_{\varphi_1}$
  - Compute its **back shifting** $I - [a, b] \cap \mathbb{T}$
  - Merge the overlapping intervals in χ_{φ}

COMBINE: *Disjunction and Eventually*

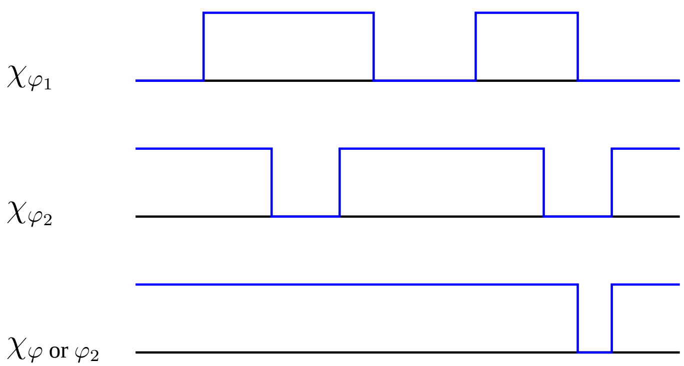
- Disjunction
  - Refine the intervals of χ_{ϕ_1} and χ_{ϕ_2} so that the mutual values of both signals become uniform in every interval
  - Compute the disjunction interval-wise
  - Merge the adjacent intervals having the same value



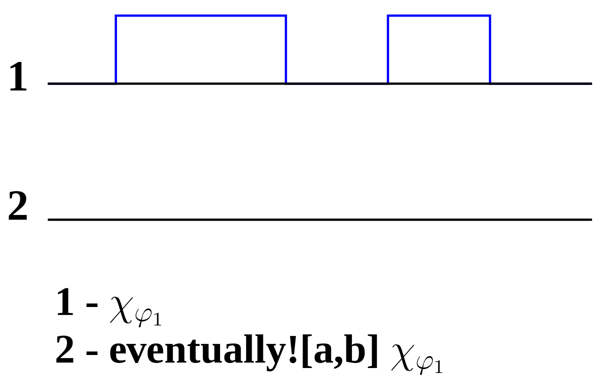
- Bounded Eventually
  - For every positive interval $I \in \chi_{\varphi_1}$
  - Compute its **back shifting** $I - [a, b] \cap \mathbb{T}$
  - Merge the overlapping intervals in χ_{φ}

Combine: *Disjunction and Eventually*

- Disjunction
  - Refine the intervals of χ_{ϕ_1} and χ_{ϕ_2} so that the mutual values of both signals become uniform in every interval
  - Compute the disjunction interval-wise
  - Merge the adjacent intervals having the same value

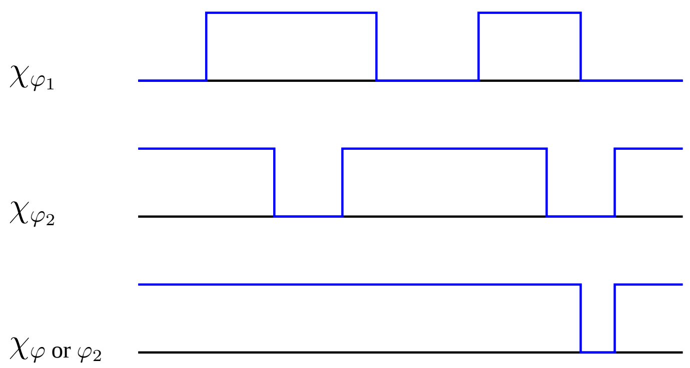


- Bounded Eventually
  - For every positive interval $I \in \chi_{\varphi_1}$
  - Compute its **back shifting** $I - [a, b] \cap \mathbb{T}$
  - Merge the overlapping intervals in χ_{φ}

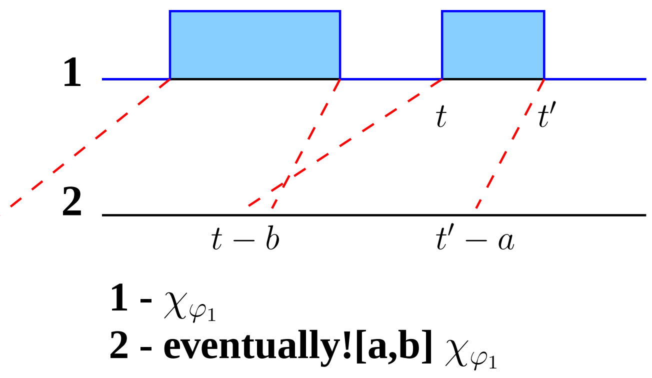


COMBINE: *Disjunction and Eventually*

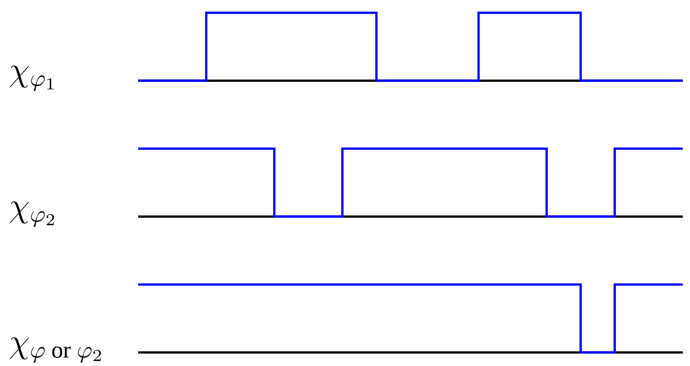
- Disjunction
  - Refine the intervals of χ_{ϕ_1} and χ_{ϕ_2} so that the mutual values of both signals become uniform in every interval
  - Compute the disjunction interval-wise
  - Merge the adjacent intervals having the same value



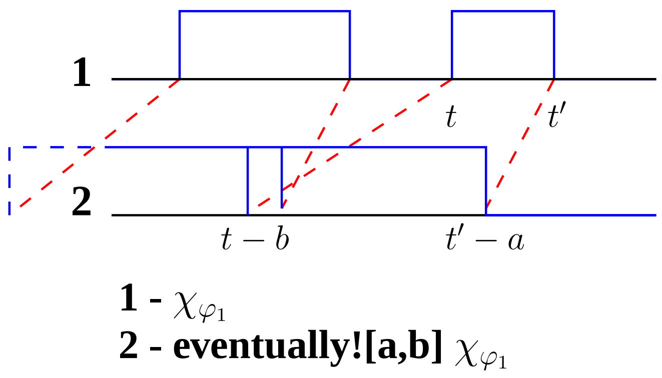
- Bounded Eventually
  - For every positive interval $I \in \chi_{\varphi_1}$
  - Compute its **back shifting** $I - [a, b] \cap \mathbb{T}$
  - Merge the overlapping intervals in χ_{φ}



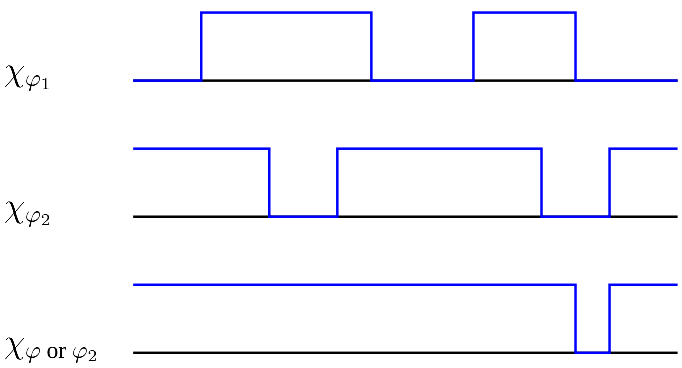
- Disjunction
  - Refine the intervals of χ_{ϕ_1} and χ_{ϕ_2} so that the mutual values of both signals become uniform in every interval
  - Compute the disjunction interval-wise
  - Merge the adjacent intervals having the same value



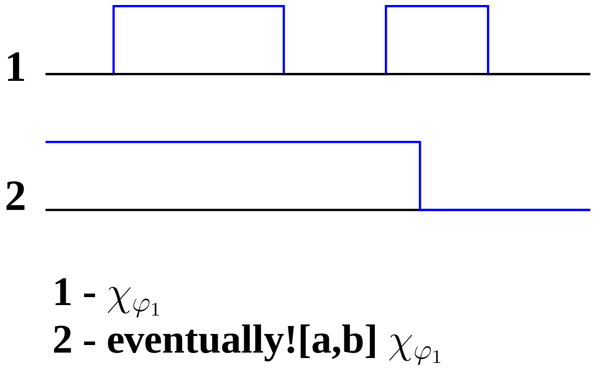
- Bounded Eventually
  - For every positive interval $I \in \chi_{\varphi_1}$
  - Compute its **back shifting** $I - [a, b] \cap \mathbb{T}$
  - Merge the overlapping intervals in χ_φ



- Disjunction
  - Refine the intervals of χ_{ϕ_1} and χ_{ϕ_2} so that the mutual values of both signals become uniform in every interval
  - Compute the disjunction interval-wise
  - Merge the adjacent intervals having the same value

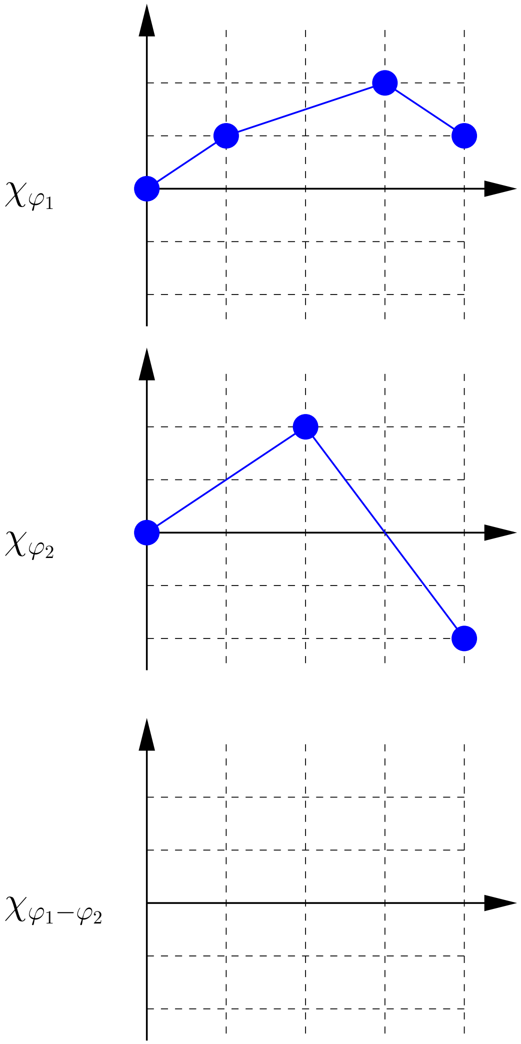


- Bounded Eventually
  - For every positive interval $I \in \chi_{\varphi_1}$
  - Compute its **back shifting** $I - [a, b] \cap \mathbb{T}$
  - Merge the overlapping intervals in χ_φ



COMBINE: *Arithmetic Operations*

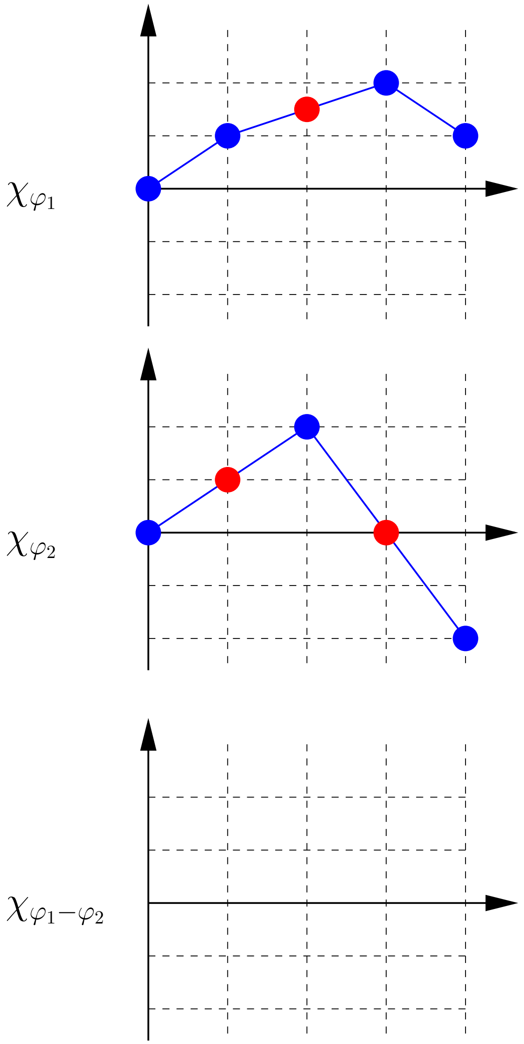
- Pointwise arithmetic operation on two signals
- Take the union of their sampling points
- Extend each signal to the new points by interpolation
- Apply the operation on each pair of sampling points



COMBINE: *Arithmetic Operations*

- Pointwise arithmetic operation on two signals
- Take the union of their sampling points

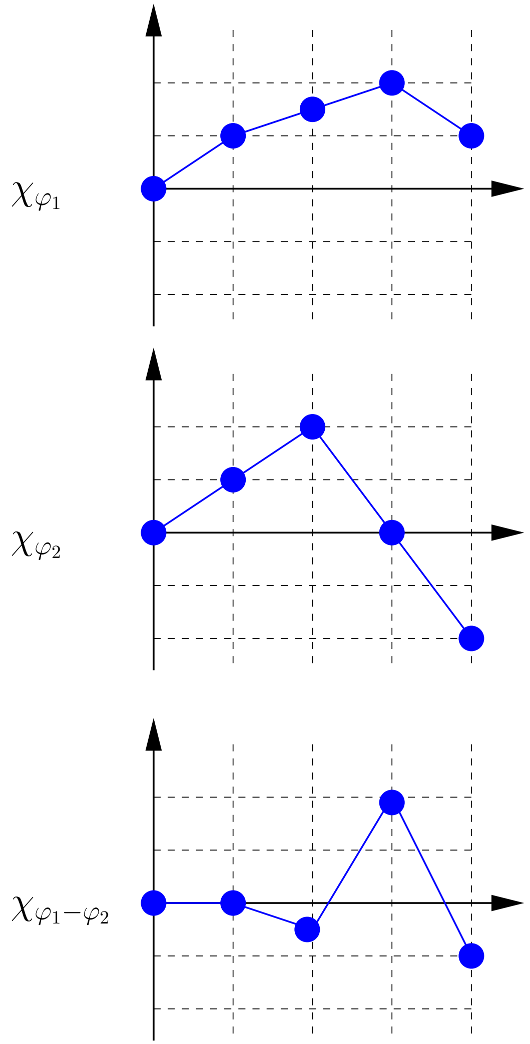
- Extend each signal to the new points by interpolation
- Apply the operation on each pair of sampling points



COMBINE: ***Arithmetic Operations***

- Pointwise arithmetic operation on two signals
- Take the union of their sampling points

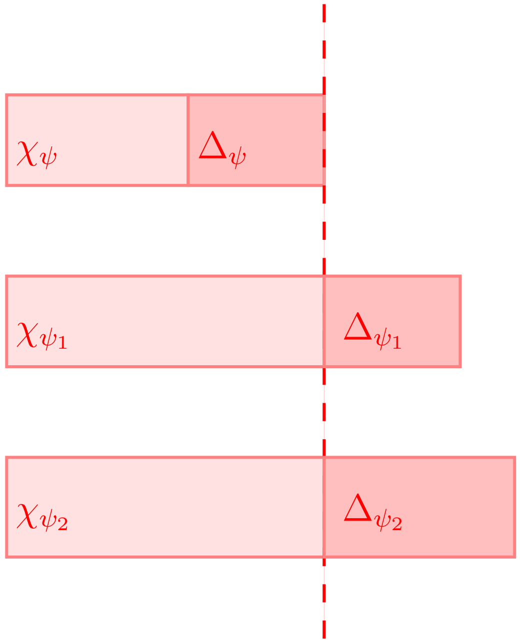
- Extend each signal to the new points by interpolation
- Apply the operation on each pair of sampling points



Incremental Marking

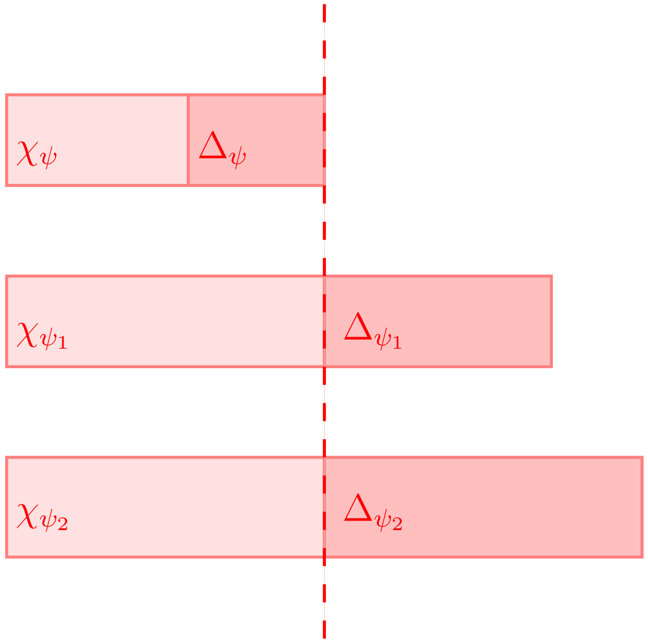
input : STL/PSL Temporal Formula φ and increment Δ_ξ

switch φ **do**
  case p
    $\Delta_\varphi := \Delta_\varphi \cdot \pi_p(\Delta_\xi)$;
  end
  case $\mathrm{OP}_2(\varphi_1, \varphi_2)$
    INCREMENTAL (φ_1, φ_2);
    $\alpha_\varphi := \mathrm{COMBINE}(\mathrm{OP}_2, \chi_{\varphi_1}, \chi_{\varphi_2}))$;
    $d := |\alpha_\varphi|$;
    $\Delta_\varphi := \Delta_\varphi \cdot \alpha_\varphi$;
    $\chi_{\varphi_1} := \chi_{\varphi_1} \cdot \langle \Delta_{\varphi_1} \rangle_d$;
    $\Delta_{\varphi_1} := d \backslash \Delta_{\varphi_1}$;
    $\chi_{\varphi_2} := \chi_{\varphi_2} \cdot \langle \Delta_{\varphi_2} \rangle_d$;
    $\Delta_{\varphi_2} := d \backslash \Delta_{\varphi_2}$
  end
end



Incremental Marking

```
input : Stl/Psl Temporal Formula φ and increment Δ_ξ
switch φ do
    case p
        Δ_φ := Δ_φ · π_p(Δ_ξ);
    end
    case op2(φ1, φ2)
        Incremental (φ1, φ2);
        α_φ := Combine(op2, χ_φ1, χ_φ2));
        d := |α_φ| ;
        Δ_φ := Δ_φ · α_φ ;
        χ_φ1 := χ_φ1 · ⟨Δ_φ1⟩_d ;
        Δ_φ1 := d\Δ_φ1 ;
        χ_φ2 := χ_φ2 · ⟨Δ_φ2⟩_d ;
        Δ_φ2 := d\Δ_φ2
    end
end
```

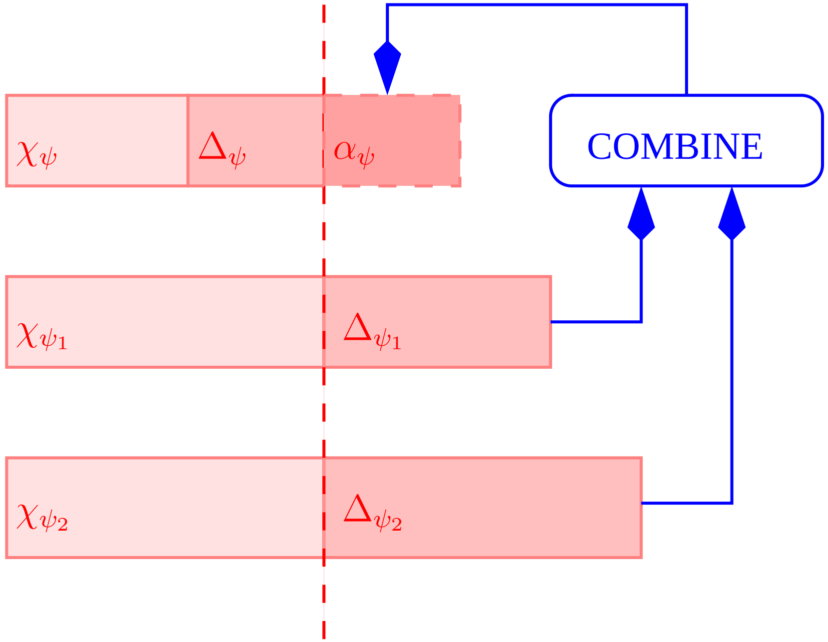


Incremental Marking

input : STL/PSL Temporal Formula φ and increment Δ_ξ

switch φ **do**
- **case** p
  - $\Delta_\varphi := \Delta_\varphi \cdot \pi_p(\Delta_\xi)$;
- **end**
- **case** $\text{OP}_2(\varphi_1, \varphi_2)$
  - INCREMENTAL (φ_1, φ_2);
  - $\alpha_\varphi :=$ COMBINE$(\text{OP}_2, \chi_{\varphi_1}, \chi_{\varphi_2}))$;
  - $d := |\alpha_\varphi|$;
  - $\Delta_\varphi := \Delta_\varphi \cdot \alpha_\varphi$;
  - $\chi_{\varphi_1} := \chi_{\varphi_1} \cdot \langle \Delta_{\varphi_1} \rangle_d$;
  - $\Delta_{\varphi_1} := d \backslash \Delta_{\varphi_1}$;
  - $\chi_{\varphi_2} := \chi_{\varphi_2} \cdot \langle \Delta_{\varphi_2} \rangle_d$;
  - $\Delta_{\varphi_2} := d \backslash \Delta_{\varphi_2}$
- **end**

end



Incremental Marking

input : STL/PSL Temporal Formula φ and increment Δ_ξ

switch φ **do**

  case p

    $\Delta_\varphi := \Delta_\varphi \cdot \pi_p(\Delta_\xi)$;

  end

  case $\text{OP}_2(\varphi_1, \varphi_2)$

    INCREMENTAL (φ_1, φ_2);

    $\alpha_\varphi := \text{COMBINE}(\text{OP}_2, \chi_{\varphi_1}, \chi_{\varphi_2}))$;

    $d := |\alpha_\varphi|$;

    $\Delta_\varphi := \Delta_\varphi \cdot \alpha_\varphi$;

    $\chi_{\varphi_1} := \chi_{\varphi_1} \cdot \langle \Delta_{\varphi_1} \rangle_d$;

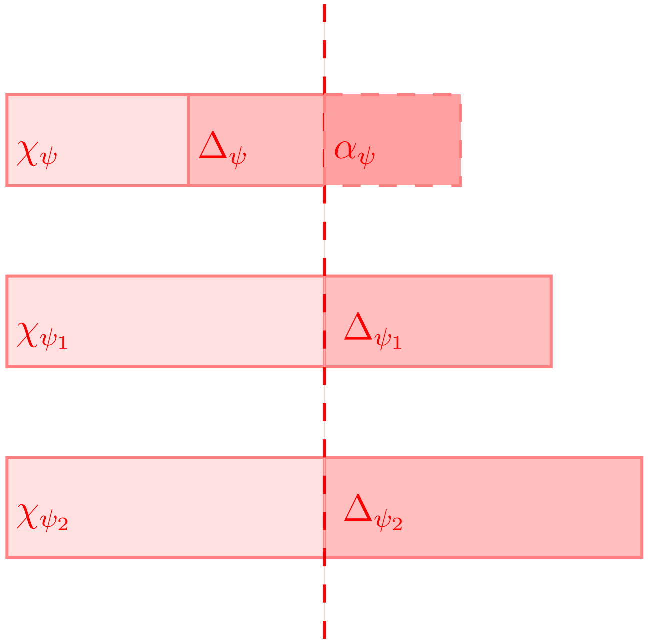
    $\Delta_{\varphi_1} := d \backslash \Delta_{\varphi_1}$;

    $\chi_{\varphi_2} := \chi_{\varphi_2} \cdot \langle \Delta_{\varphi_2} \rangle_d$;

    $\Delta_{\varphi_2} := d \backslash \Delta_{\varphi_2}$

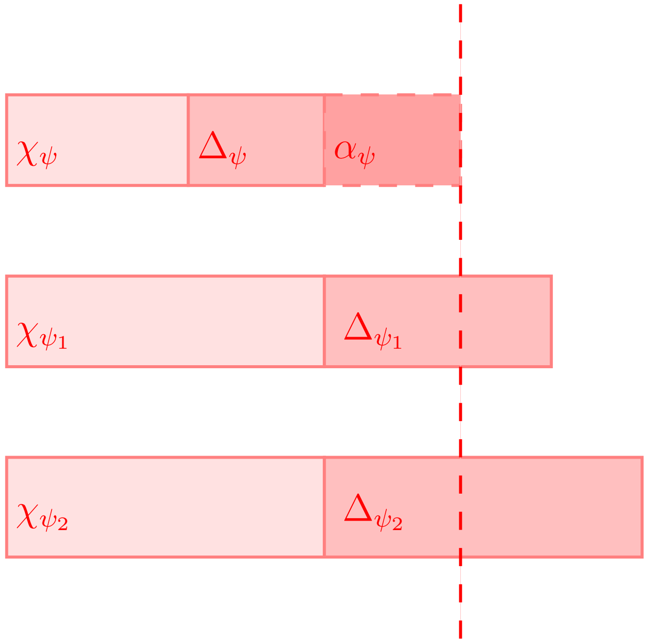
  end

end



input : STL/PSL Temporal Formula φ and increment Δ_ξ

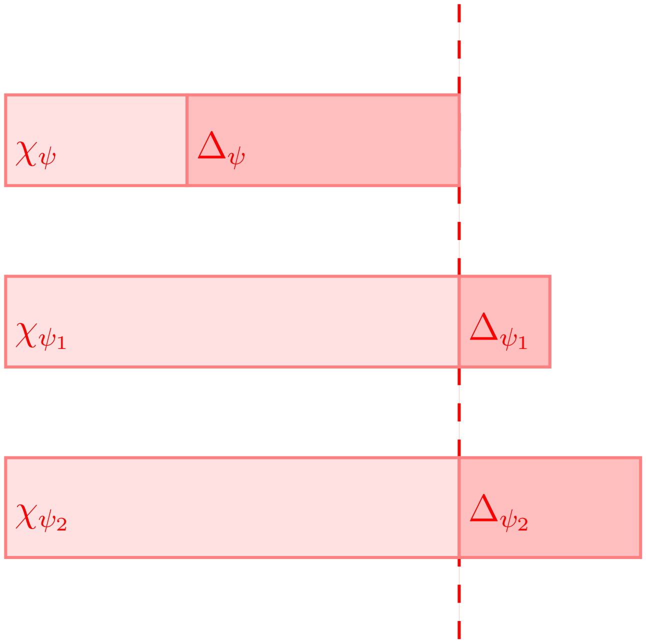
switch φ **do**
  case p
    $\Delta_\varphi := \Delta_\varphi \cdot \pi_p(\Delta_\xi)$;
  end
  case $\text{OP}_2(\varphi_1, \varphi_2)$
    INCREMENTAL (φ_1, φ_2);
    $\alpha_\varphi := \text{COMBINE}(\text{OP}_2, \chi_{\varphi_1}, \chi_{\varphi_2}))$;
    $d := |\alpha_\varphi|$;
    $\Delta_\varphi := \Delta_\varphi \cdot \alpha_\varphi$;
    $\chi_{\varphi_1} := \chi_{\varphi_1} \cdot \langle \Delta_{\varphi_1} \rangle_d$;
    $\Delta_{\varphi_1} := d \backslash \Delta_{\varphi_1}$;
    $\chi_{\varphi_2} := \chi_{\varphi_2} \cdot \langle \Delta_{\varphi_2} \rangle_d$;
    $\Delta_{\varphi_2} := d \backslash \Delta_{\varphi_2}$
  end
end



Incremental Marking

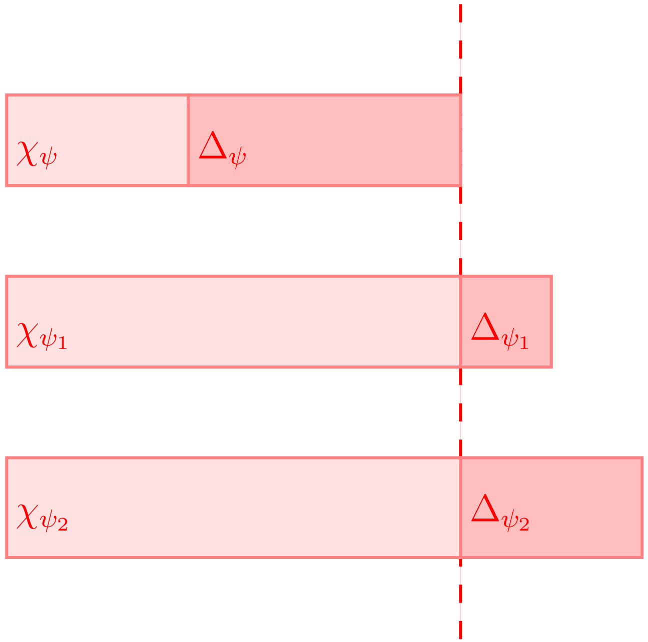
input : STL/PSL Temporal Formula φ and increment Δ_ξ

switch φ **do**
  case p
    $\Delta_\varphi := \Delta_\varphi \cdot \pi_p(\Delta_\xi)$;
  end
  case $\text{OP}_2(\varphi_1, \varphi_2)$
    INCREMENTAL (φ_1, φ_2);
    $\alpha_\varphi := \text{COMBINE}(\text{OP}_2, \chi_{\varphi_1}, \chi_{\varphi_2}))$;
    $d := |\alpha_\varphi|$;
    $\Delta_\varphi := \Delta_\varphi \cdot \alpha_\varphi$;
    $\chi_{\varphi_1} := \chi_{\varphi_1} \cdot \langle \Delta_{\varphi_1} \rangle_d$;
    $\Delta_{\varphi_1} := d \backslash \Delta_{\varphi_1}$;
    $\chi_{\varphi_2} := \chi_{\varphi_2} \cdot \langle \Delta_{\varphi_2} \rangle_d$;
    $\Delta_{\varphi_2} := d \backslash \Delta_{\varphi_2}$
  end
end



Incremental Marking

- Advantages of the incremental algorithm
  - Often more memory efficient
    - Determined parts of the signal may be discarded
  - Early detection of errors



AMT *Tool Overview*

- Stand alone tool for lightweight verification of properties on continuous signals
- **Inputs:**
  - STL/PSL specification
  - Input signals (Boolan or continuous)
    - From a file (**raw**, **vcd** and **out** format)
    - Dynamic inputs through TCP/IP packets
- **Property evaluation:**
  - Offline
  - Incremental
- Visual evaluation of results

AMT *Tool Overview*

- Stand alone tool for lightweight verification of properties on continuous signals
- **Inputs:**
  - STL/PSL specification
  - Input signals (Boolan or continuous)
    - From a file (**raw**, **vcd** and **out** format)
    - Dynamic inputs through TCP/IP packets
- **Property evaluation:**
  - Offline
  - Incremental
- Visual evaluation of results

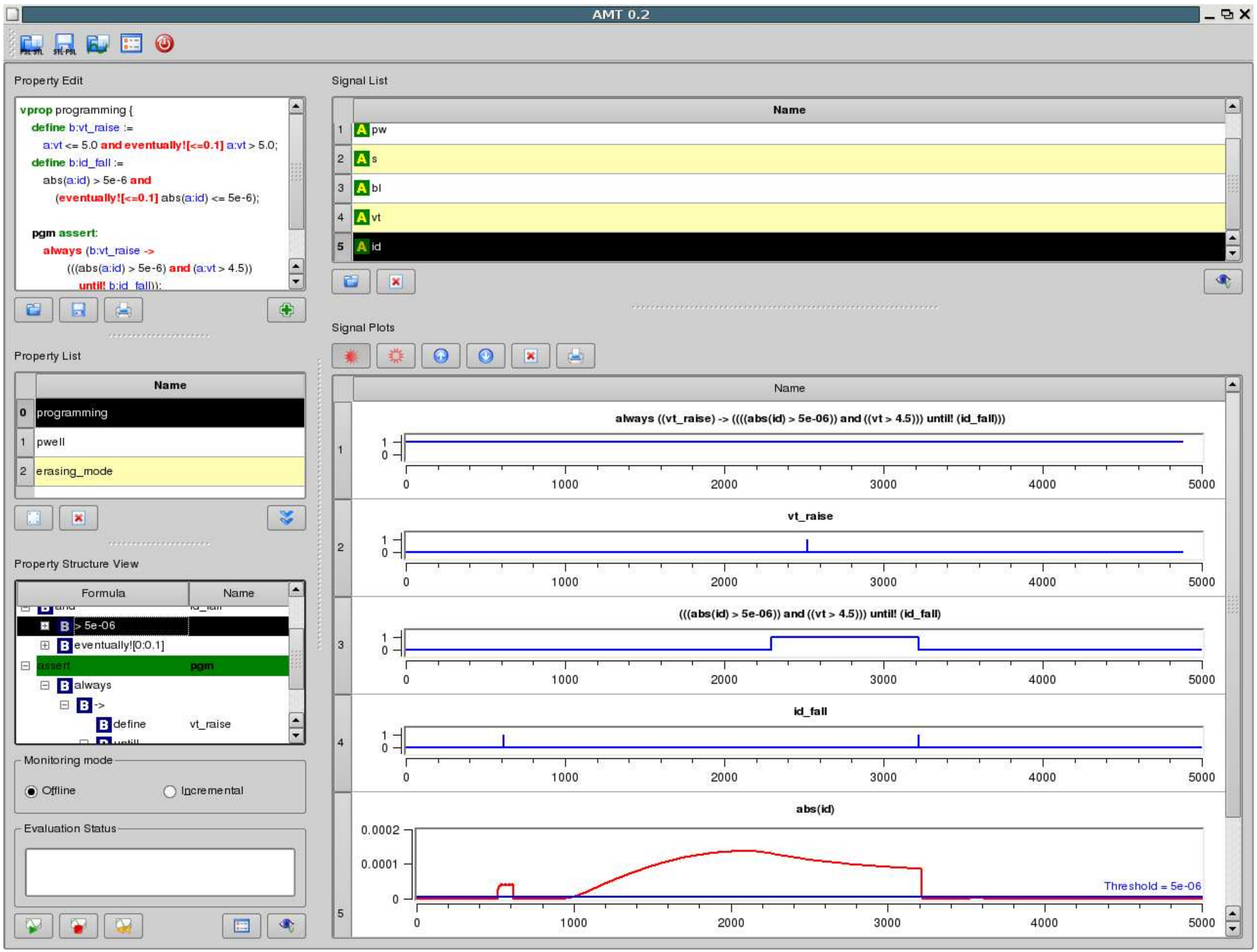
AMT *Tool Overview*

- Stand alone tool for lightweight verification of properties on continuous signals
- **Inputs:**
  - STL/PSL specification
  - Input signals (Boolan or continuous)
    - From a file (**raw**, **vcd** and **out** format)
    - Dynamic inputs through TCP/IP packets
- **Property evaluation:**
  - Offline
  - Incremental
- Visual evaluation of results

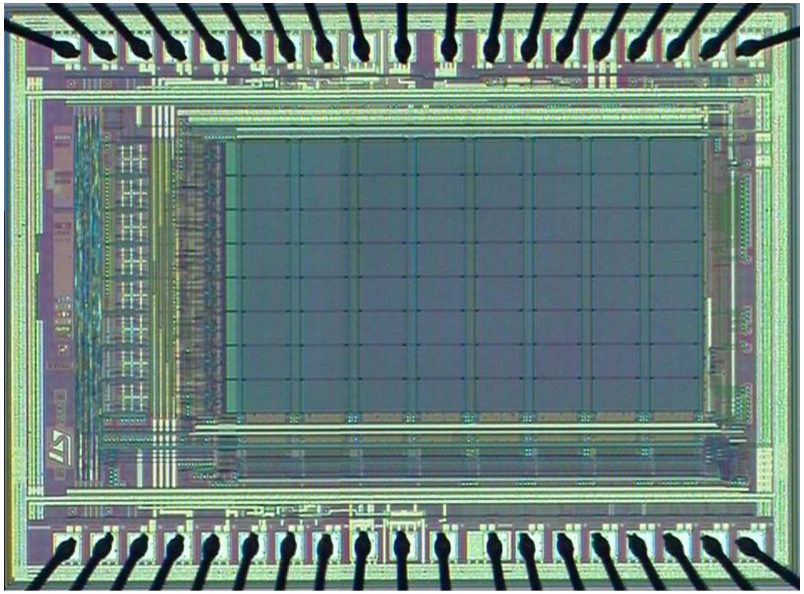
AMT *Tool Overview*

- Stand alone tool for lightweight verification of properties on continuous signals
- **Inputs:**
  - STL/PSL specification
  - Input signals (Boolan or continuous)
    - From a file (**raw**, **vcd** and **out** format)
    - Dynamic inputs through TCP/IP packets
- **Property evaluation:**
  - Offline
  - Incremental
- Visual evaluation of results

AMT *Tool: Main Window*



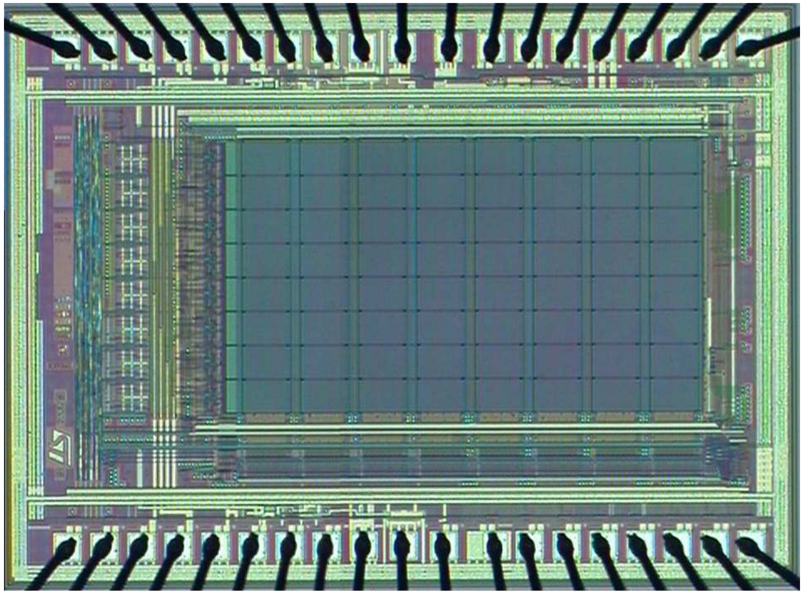
FLASH *Memory Case Study*



- Provided by STM Italy
- Why Flash memory?
  - **Analog** circuit that implements **digital** behavior
  - Good connection between **analog** and **digital** worlds

- Different modes
  - Programming, reading, erasing, etc.
- Characteristic signals
  - **bl**: bit line terminal
  - **pw**: p-well terminal
  - **wl**: word line
  - **s**: source terminal
  - **vt**: threshold voltage of cell
  - **id**: drain current of cell
- Correct functioning in a given mode determined by the behavior of the characteristic signals
- 5 properties specifying the correct behavior

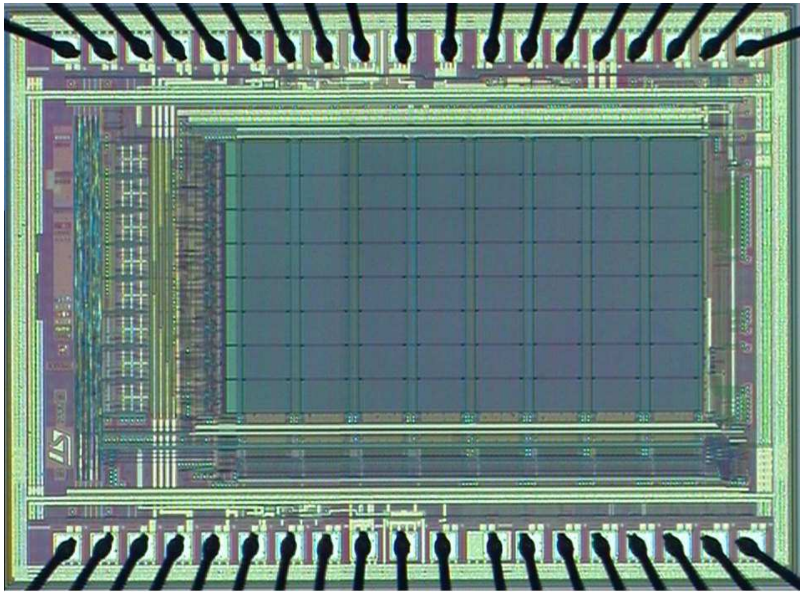
FLASH *Memory Case Study*



- Provided by STM Italy
- Why Flash memory?
  - **Analog** circuit that implements **digital** behavior
  - Good connection between **analog** and **digital** worlds

- Different modes
  - Programming, reading, erasing, etc.
- Characteristic signals
  - **bl**: bit line terminal
  - **pw**: p-well terminal
  - **wl**: word line
  - **s**: source terminal
  - **vt**: threshold voltage of cell
  - **id**: drain current of cell
- Correct functioning in a given mode determined by the behavior of the characteristic signals
- 5 properties specifying the correct behavior

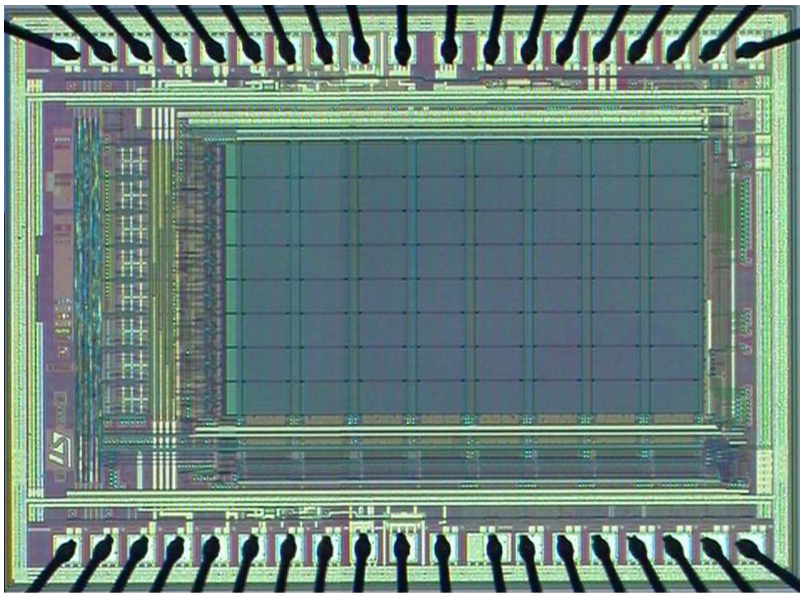
FLASH *Memory Case Study*



- Provided by STM Italy
- Why Flash memory?
  - **Analog** circuit that implements **digital** behavior
  - Good connection between **analog** and **digital** worlds

- Different modes
  - Programming, reading, erasing, etc.
- Characteristic signals
  - **bl**: bit line terminal
  - **pw**: p-well terminal
  - **wl**: word line
  - **s**: source terminal
  - **vt**: threshold voltage of cell
  - **id**: drain current of cell
- Correct functioning in a given mode determined by the behavior of the characteristic signals
- 5 properties specifying the correct behavior

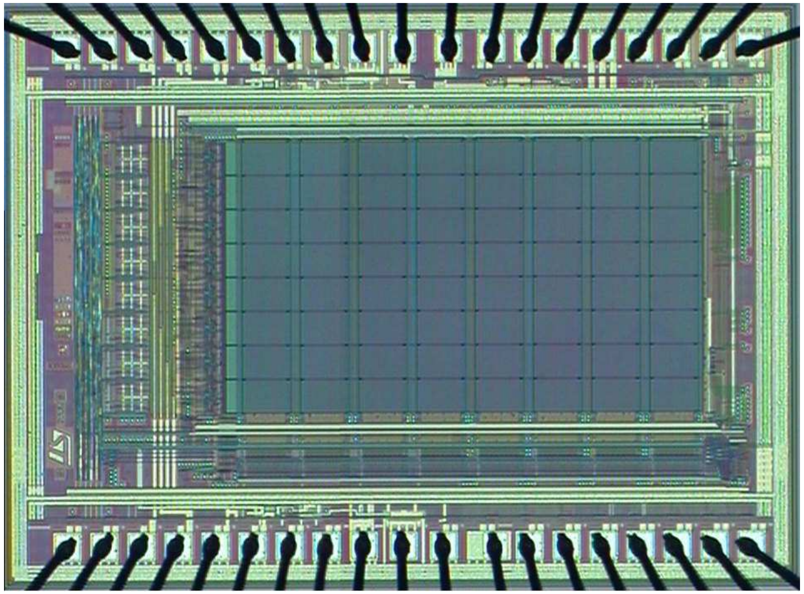
FLASH *Memory Case Study*



- Provided by STM Italy
- Why Flash memory?
  - **Analog** circuit that implements **digital** behavior
  - Good connection between **analog** and **digital** worlds

- Different modes
  - Programming, reading, erasing, etc.
- Characteristic signals
  - **bl**: bit line terminal
  - **pw**: p-well terminal
  - **wl**: word line
  - **s**: source terminal
  - **vt**: threshold voltage of cell
  - **id**: drain current of cell
- Correct functioning in a given mode determined by the behavior of the characteristic signals
- 5 properties specifying the correct behavior

FLASH *Memory Case Study*



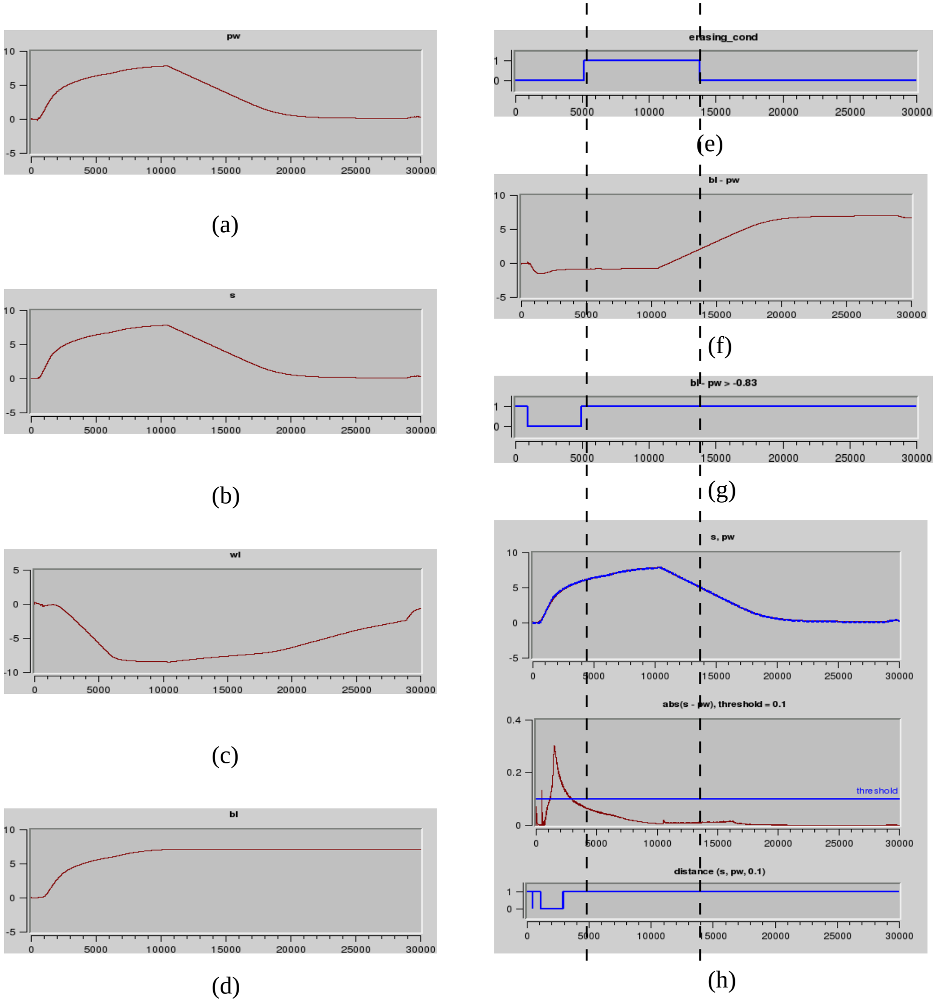
- Provided by STM Italy
- Why Flash memory?
  - **Analog** circuit that implements **digital** behavior
  - Good connection between **analog** and **digital** worlds

- Different modes
  - Programming, reading, erasing, etc.
- Characteristic signals
  - **bl**: bit line terminal
  - **pw**: p-well terminal
  - **wl**: word line
  - **s**: source terminal
  - **vt**: threshold voltage of cell
  - **id**: drain current of cell
- Correct functioning in a given mode determined by the behavior of the characteristic signals
- 5 properties specifying the correct behavior

Case Study Example: Erasing Property

```
vprop erasing {
  define b:erasing_cond :=
    a:wl <= -6 and a:pw > 5;

  erasing assert:
    always (b:erasing_cond ->
      (distance (a:s,a:pw,0.1)
      and (a:bl-a:pw)>-0.83));
}
```



Tool Evaluation

name	pgm sim # intervals	erase sim # intervals
wl	34829	283624
pw	25478	283037
s	33433	282507
bl	32471	139511
id	375	n/a

Table 1: Input Size

Tool Evaluation

name	pgm sim # intervals	erase sim # intervals
wl	34829	283624
pw	25478	283037
s	33433	282507
bl	32471	139511
id	375	n/a

Table 1: Input Size

property	time (s)	# intervals
programming1	0.14	99715
programming2	0.42	405907
p-well	0.12	89071
decay	0.50	594709
erasing	2.35	2968578

Table 2: Offline Algorithm Evaluation

Tool Evaluation

name	pgm sim # intervals	erase sim # intervals
wl	34829	283624
pw	25478	283037
s	33433	282507
bl	32471	139511
id	375	n/a

Table 1: Input Size

property	time (s)	# intervals
programming1	0.14	99715
programming2	0.42	405907
p-well	0.12	89071
decay	0.50	594709
erasing	2.35	2968578

Table 2: Offline Algorithm Evaluation

Property	Offline t = total # intervals	Incremental m = max # active intervals	m/t * 100
programming1	99715	65700	65.9
programming2	594709	242528	40.8
p-well	89071	8	0.01
decay	594709	279782	47.1

Table 3: Offline/Incremental Space Requirement Comparison

Conclusion

- **Main contributions:**
  - AMT tool that monitors temporal properties of continuous signals
    - Description of properties in STL/PSL specification language
    - Offline and incremental algorithms
    - Integration with numerical simulatiors via simulation dump files or TCP/IP link
  - FLASH memory case study
    - Validates the tool and the approach
    - Shows the automation and efficiency of monitoring continuous signals

Conclusion

- **Main contributions:**
  - AMT tool that monitors temporal properties of continuous signals
    - Description of properties in STL/PSL specification language
    - Offline and incremental algorithms
    - Integration with numerical simulatiors via simulation dump files or TCP/IP link
  - FLASH memory case study
    - Validates the tool and the approach
    - Shows the automation and efficiency of monitoring continuous signals