Temporal Logics and Semantics

Stefano Tonetta

Embedded System Unit, Fondazione Bruno Kessler

ETCS Workshop
Trento October 9, 2008
Outline

1. Models and constraints
2. Static formulas
   - Syntax
   - Semantics
3. Temporal formulas
   - Syntax
   - Semantics
4. Controlled natural language
5. Real-time and Hybrid models
6. Methods, state machines, and interleaving semantics
Outline

1. Models and constraints
   - Static formulas
     - Syntax
     - Semantics
   - Temporal formulas
     - Syntax
     - Semantics
   - Controlled natural language
   - Real-time and Hybrid models
   - Methods, state machines, and interleaving semantics
Formal class diagrams

Definition

A class diagram consists of:

**Primitive Types**  A finite set of primitive types $\mathcal{T} = \{\tau_1, \ldots, \tau_h\}$.

**Classes**  A finite set of classes $C = \{c_1, \ldots, c_n\}$.

**Attributes**  For each class $c \in C$, a finite set of attributes $c.A = \{c.a_1, \ldots, c.a_m\}$. Every attribute $c.a$ has

- a type $c.a.TYPE \in C \cup \mathcal{T}$;
- a multiplicity $c.a.MULTIPLICITY$ that defines a bounded integer range $n..m$, with $0 \leq n \leq m$. 
Definition

An object model is an interpretation of all functional symbols of a class diagram. I.e., for all class \( c \in C \), the object model defines the objects of class \( c \), and, for all objects \( o \) of class \( c \), for all attributes/associations \( a \) of \( c \), the object model assigns a (type-consistent) value to \( o.a \).
From class diagram to object models

Example

An object model assigns to each balise group a position (for example 15), a set of balises (for example \{b_1, b_2, b_3\}), a message (for example 001110...).
A static constraint specifies if an object model is bad or good.
Static constraint

A static constraint specifies if an object model is bad or good.

**Definition**

A static constraint is a set of object models.
A static constraint specifies if an object model is bad or good.

**Definition**

A static constraint is a set of object models.
A static constraint specifies if an object model is bad or good.

**Definition**

A static constraint is a set of object models.
A behavior is an (infinite) sequence of object models.
A behavior is an (infinite) sequence of object models.
Definition

A behavior is an (infinite) sequence of object models.
A temporal constraint specifies if a behavior is bad or good.
Temporal constraint

A temporal constraint specifies if a behavior is bad or good.

**Definition**

A temporal constraint is a set of behavior.
A temporal constraint specifies if a behavior is bad or good.

**Definition**

A temporal constraint is a set of behavior.
Outline

1. Models and constraints
2. Static formulas
   - Syntax
   - Semantics
3. Temporal formulas
   - Syntax
   - Semantics
4. Controlled natural language
5. Real-time and Hybrid models
6. Methods, state machines, and interleaving semantics
Constraints as formulas

- We use formulas to represent static and temporal constraints.
- We formally define syntax and semantics.
- **Syntax**: how a formula can be built.
- **Semantics**: what the formula means.
- Static formulas are a fragment of first-order logic.
- Temporal formulas are a fragment of first-order temporal logic.
Outline

1. Models and constraints
2. Static formulas
   - Syntax
   - Semantics
3. Temporal formulas
   - Syntax
   - Semantics
4. Controlled natural language
5. Real-time and Hybrid models
6. Methods, state machines, and interleaving semantics
\[ ma_1 . \text{balises}.size = 2 \land ma\.balises[2] = ma\.balises[1] + 15 \]

\[ ma_1 . \text{balises}.size = 2 \quad ma_1 . \text{balises}[2] = ma_1 . \text{balises}[1] + 15 \]

\[ ma_1 . \text{balises}.size \]

\[ ma_1 . \text{balises}[2] \quad ma_1 . \text{balises}[1] \]

\[ ma_1 \quad 2 \quad 1 \quad 15 \]

\[ ma_1 . \text{balises} \quad \text{size} \quad 2 \quad \land \quad 1 \quad [ \quad ] \quad 15 \quad \text{SYMBOLS} \]
Vocabulary

- Syntax depends on the class diagram.
- We assume to have constants, functions, and predicates for primitive symbols (example: $0, 1, 2, TRUE, +, *, >, \leq, ...$).
- We assume a given set of variables $V_\tau$ for each type $\tau \in C \cup T$. 
Terms may be:

- Variables;
- Constants;
- Functions: $f(t_1, \ldots, t_n)$, where $t_1, \ldots, t_n$ are terms;
- Attributes: $t.a$, where $t$ is a term;
- Arrays: $t.a[i]$ and $t.a.size$, where $t$ is a term and $a$ has max multiplicity $> 1$.

Example

$x$
$x + y$
$\text{train.position}$
$\text{train.position} + y$
...

Stefano Tonetta (FBK)  TL & Semantics  October 9, 2008  15 / 47
Static formulas

Formulas may be:

- **Relations**: $R(t_1, \ldots, t_n)$, where $t_1, \ldots, t_n$ are terms;
- **Comparisons**: $t_1 = t_2$, where $t_1$ and $t_2$ are terms;
- **Collection membership**: $t_1 \in t_2$, where $t_1$ and $t_2$ are terms;
- **Boolean Combinations**: $\neg \varphi_1$ and $\varphi_1 \land \varphi_2$: where $\varphi_1$ and $\varphi_2$ are formulas;
- **Quantifiers**: $\forall v : \tau. \varphi$ and $\forall v \in t. \varphi$, where $\varphi$ is a formula, $v$ a variable, $t$ a term.

**Example**

$x > 10$

$train_1.current\_section \in train_1.ma.sections$

$\forall t_1 : Train. \forall t_2 : Train. (t_1.position = t_2.position)$

...
Outline

1. Models and constraints
2. Static formulas
   - Syntax
   - Semantics
3. Temporal formulas
   - Syntax
   - Semantics
4. Controlled natural language
5. Real-time and Hybrid models
6. Methods, state machines, and interleaving semantics
Interpretation of terms

- Interpretation given by an object model $M$, and an assignment $\mu$ to variables.
- Extended to all terms: for example, if $\mathcal{I}(t) = train_1$ and $\mathcal{I}(position)(train_1) = 15$, then $[[t.a]] = 15$. 
Interpretation of terms

Formally,

Variables \([v]_{M,\mu} = \mu(v)\).

Constants \([c]_{M,\mu} = I(c)\).

Functions \([f(t_1, \ldots, t_n)]_{M,\mu} = I(f)([t_1]_{M,\mu} \cdots [t_n]_{M,\mu})\).

Simple Attributes If \(I(a)([t]_{M,\mu}) = [q]\), then \([t.a]_{M,\mu} = q\).

Multiple Attributes \([t.a]_{M,\mu} = I(a)([t]_{M,\mu})\)
\([t.a.size]_{M,\mu} = |I(a)([t]_{M,\mu})|\).
Models of static formulas

- An object model satisfies a formula if the interpretation of the terms makes the formula true.
Models of static formulas

- Formally,

  Relations \( \langle M, \mu \rangle \models R(t_1, \ldots, t_n) \iff \mathcal{I}(R)([[t_1]]_{\langle M, \mu \rangle} \cdots [[t_n]]_{\langle M, \mu \rangle}) \) holds.

  Comparisons \( \langle M, \mu \rangle \models t_1 = t_2 \iff [[t_1]]_{\langle M, \mu \rangle} = [[t_2]]_{\langle M, \mu \rangle} \).

  Collection membership \( \langle \omega, \mu \rangle \models t_1 \in t_2 \iff \text{for some } i, 1 \leq i \leq |[[t_2]]_{\langle \omega, \mu \rangle}|, [[t_1]]_{\langle \omega, \mu \rangle} = [[t_2[i]]]_{\langle \omega, \mu \rangle} \).

  Boolean Combinations \( \langle M, \mu \rangle \models \neg \varphi_1 \iff \langle M, \mu \rangle \not\models \varphi_1 \),

  \( \langle M, \mu \rangle \models \varphi_1 \land \varphi_2 \iff \langle M, \mu \rangle \models \varphi_1 \) and \( \langle M, \mu \rangle \models \varphi_2 \).

  Quantifiers \( \langle M, \mu \rangle \models \forall v \in t.\varphi \iff \text{for all } q \in [[t]]_{\langle M, \mu \rangle}, \langle M, \mu[q/v] \rangle \models \varphi \).

- The models of a closed formula are all object models that satisfy the formula.

- Thus, a static formula represents a static constraint.
Outline

1. Models and constraints
2. Static formulas
   - Syntax
   - Semantics
3. Temporal formulas
   - Syntax
   - Semantics
4. Controlled natural language
5. Real-time and Hybrid models
6. Methods, state machines, and interleaving semantics
1. Models and constraints
2. Static formulas
   - Syntax
   - Semantics
3. Temporal formulas
   - Syntax
   - Semantics
4. Controlled natural language
5. Real-time and Hybrid models
6. Methods, state machines, and interleaving semantics
Temporal terms and expressions

**Definition**
A temporal term of type $\tau$ either is an $L_D$ term of type $\tau$ or is an expression $\text{next}(t)$, where $t$ is a $L_D$ term of type $\tau$.

**Definition**
Transition expressions are like static formulas but may contain the $\text{next}$ operator.
Regular expressions

Transition expressions Any transition expression is a regular expression.

Empty word $\epsilon$ is a regular expression.

Regular Operators If $r_1$ and $r_2$ are regular expressions, then $r_1^*$, $r_1; r_2$, $r_1 : r_2$, $r_1 | r_2$, $r_1 && r_2$ are regular expressions.
Formulas

Transition expressions Any transition expression is a formula.

Boolean Combinations If $\varphi_1$ and $\varphi_2$ are formulas, then $\neg \varphi_1$, $\varphi_1 \land \varphi_2$ are formulas.

Temporal Operators If $\varphi_1$ and $\varphi_2$ are formulas, then $X \varphi_1$, $\varphi_1 U \varphi_2$ are formulas.

Suffix Operators If $r$ is a regular expression and $\varphi$ is a linear temporal formula, then $\{r\}\varphi$ is a formula.
Outline

1. Models and constraints

2. Static formulas
   - Syntax
   - Semantics

3. Temporal formulas
   - Syntax
   - Semantics

4. Controlled natural language

5. Real-time and Hybrid models

6. Methods, state machines, and interleaving semantics
Temporal terms

**Term** A term is interpreted as in the static version.

**Next Term** A next term \( \text{next}(t) \) is interpreted at the step \( i \) with the value of \( t \) at step \( i + 1 \).

**Transition Expressions** The models of transition expressions are defined as for the static formulas.

\[
v = \text{next}(v)
\]
Regular expressions

\[ \epsilon \]

\[ r^* \]

\[ r \]

\[ r_1; r_2 \]

\[ r_1 \]

\[ r_2 \]
Regular expressions

\[ r_1 : r_2 \]

\[ r_1 \mid r_2 \]

\[ r_1 \land r_2 \]

\[ r_1, r_2 \]
Regular expressions

\[ \langle \omega, \mu \rangle \models^{i \cdots j} \epsilon \text{ iff } i = j. \]
\[ \langle \omega, \mu \rangle \models^{i \cdots j} r^* \text{ iff } i = j \text{ or there exists } k, i < k \leq j, \]
\[ \langle \omega, \mu \rangle \models^{i \cdots k} r, \langle \omega, \mu \rangle \models^{k \cdots j} r^*; \]
\[ \langle \omega, \mu \rangle \models^{i \cdots j} r_1; r_2 \text{ iff there exists } k, i \leq k \leq j, \]
\[ \langle \omega, \mu \rangle \models^{i \cdots k} r_1, \langle \omega, \mu \rangle \models^{k \cdots j} r_2; \]
\[ \langle \omega, \mu \rangle \models^{i \cdots j} r_1 : r_2 \text{ iff there exists } k, i < k \leq j, \]
\[ \langle \omega, \mu \rangle \models^{i \cdots k} r_1, \langle \omega, \mu \rangle \models^{k-1 \cdots j} r_2; \]
\[ \langle \omega, \mu \rangle \models^{i \cdots j} r_1 | r_2 \text{ iff } \langle \omega, \mu \rangle \models^{i \cdots j} r_1 \text{ or } \langle \omega, \mu \rangle \models^{i \cdots j} r_2; \]
\[ \langle \omega, \mu \rangle \models^{i \cdots j} r_1 \& \& r_2 \text{ iff } \langle \omega, \mu \rangle \models^{i \cdots j} r_1 \text{ and } \langle \omega, \mu \rangle \models^{i \cdots j} r_2. \]
Temporal formulas

$X \varphi$

$\varphi_1 U \varphi_2$

$\{ r \} \varphi$

$r \varphi$

Stefano Tonetta (FBK)
Temporal formulas

Boolean $\langle \omega, \mu \rangle \models \neg \varphi_1$ iff $\langle \omega, \mu \rangle \not\models \varphi_1$, $\langle \omega, \mu \rangle \models \varphi_1 \land \varphi_2$ iff $\langle \omega, \mu \rangle \models \varphi_1$ and $\langle \omega, \mu \rangle \models \varphi_2$.

Temporal Operators
$\langle \omega, \mu \rangle \models X \varphi$ iff $\langle \omega^1, \mu \rangle \models \varphi$;
$\langle \omega, \mu \rangle \models \varphi_1 U \varphi_2$ iff there exists $i \geq 0$ such that $\langle \omega^i, \mu \rangle \models \varphi_2$ and for all $0 \leq j < i$ $\langle \omega^j, \mu \rangle \models \varphi_1$.

Suffix Operators $\langle \omega, \mu \rangle \models \{ r \} \varphi$ iff there exists $i \geq 0$ such that $\langle \omega^i, \mu \rangle \models \varphi$ and $\langle \omega, \mu \rangle \models^0..i+1 r$. 
Definition

Given a formula $\varphi$, the satisfiability problem consists of finding a sequence of models $\omega$, and an assignment $\mu$ to the free variables of $\varphi$, such that $\langle \omega, \mu \rangle \models \varphi$. 

Formula $\xrightarrow{\text{SAT}}$ Model
Outline

1. Models and constraints
2. Static formulas
   - Syntax
   - Semantics
3. Temporal formulas
   - Syntax
   - Semantics
4. Controlled natural language
5. Real-time and Hybrid models
6. Methods, state machines, and interleaving semantics
Standard abbreviations

- $\varphi_1 \lor \varphi_2 \equiv \neg(\neg\varphi_1 \land \neg\varphi_2)$;
- $\varphi_1 \rightarrow \varphi_2 \equiv \neg\varphi_1 \lor \varphi_2$;
- $\exists x \in t.(\varphi) \equiv \neg\forall x \in t.(\neg\varphi)$;
- $t_1 \notin t_2 \equiv \neg(t_1 \in t_2)$;
- $F \varphi \equiv \top \cup \varphi$;
- $G \varphi \equiv \neg F \neg \varphi$;
- $r \mid\rightarrow \varphi \equiv \neg\{r\}\neg \varphi$.

When we specify the constraints of a particular class $c \in C$, we implicitly specify the constraint for every object $o$ of that class, the constraint holds.

In this case, we can use a meaning $o.a.$
English sugaring

We consider the following English expressions as synonyms:

- “always” \( f \equiv Gf \)
- “never” \( f \equiv G\neg(f) \)
- “in the future” \( f \equiv F(f) \)
- “until” \( f_1 \quad “until” \quad f_2 \equiv f_1 Uf_2 \)
- “infinitely many times” \( f \equiv GF(f) \)
- “will eventually hold” \( f \equiv F(f) \)
- “every time” \( f_1 \quad “holds,” \quad f_2 \equiv G(f_1 \to f_2) \)
- “not” \( f \equiv \neg f \)
- “and” \( f_1 \quad “and” \quad f_2 \equiv f_1 \land f_2 \)
- “or” \( f_1 \quad “or” \quad f_2 \equiv f_1 \lor f_2 \)
- “implies” \( f_2 \equiv f_1 \to f_2 \)
- “if” \( f_1 \quad “then” \quad f_2 \quad “else” \quad f_3 \equiv (f_1 \to f_2) \land (\neg f_1 \to f_3) \)
- “for all” \( s \times f, \quad f \equiv \forall x : s(f) \)
- “there exists a” \( s \times \quad “such that” \quad f \equiv \exists x : s(f) \)
- “for all” \( x \quad “in” \quad e, \quad f \equiv \forall x \in e(f) \)
- “there exists” \( x \quad “in” \quad e \quad “such that” \quad f \equiv \exists x \in e(f) \)
**Outline**

1. Models and constraints
2. Static formulas
   - Syntax
   - Semantics
3. Temporal formulas
   - Syntax
   - Semantics
4. Controlled natural language
5. Real-time and Hybrid models
6. Methods, state machines, and interleaving semantics
So far, models have been sequences of object models.
Timed and hybrid sequences

- So far, models have been sequences of object models.
- We can deal also with timed sequences.
Timed and hybrid sequences

- So far, models have been sequences of object models.
- We can deal also with timed sequences.
- Further step: hybrid sequences:
  - time is weakly monotonic;
  - there are discrete (modes) and continuous variables (timers, dynamics);
  - there are two kind of steps (phases):
    - discrete step: not time elapse, mode change, timer resets;
    - continuous step: time elapses, no mode change, continuous evolution.
Continuous variables

- No existing hybrid semantics for the logic.
- Proposed model: dense, weakly monotonic time, with discrete observations, and predicates over derivatives of linear evolution.
- Good compromise between precision and tractability.
- Example: “in the future der(train.position) > 300”
1 Models and constraints
2 Static formulas
   • Syntax
   • Semantics
3 Temporal formulas
   • Syntax
   • Semantics
4 Controlled natural language
5 Real-time and Hybrid models
6 Methods, state machines, and interleaving semantics
Semantics of methods and state machines

- Semantics tailor-made for ETCS.
- The semantics of a method is given by:
  - an event \textit{start} that is true when the method is activated;
  - a condition \textit{end} that is true when the method has terminated.
- The semantics of a state machine is given by a variable \textit{state} whose domain is the set of states of the state machine.
- We consider only models that satisfy the following conditions:
  - Initially a method is not active.
  - A method can be called only if it is not active.
  - If a state machine is associated to a method, the state changes according to the transitions of the state machine.
Terms for methods and state machines

Methods \( o.m() \)
- true when the method is called

Methods with parameters \( o.m(o_1, \ldots, o_k) \)
- true when the method is called with the given parameters

Methods with return \( o.m() \) returns \( r \)
- true when the method returns a given value.

States (term) \( o.m.State \)
- evaluates to the current state of the state machine of the method.
Asynchronous interleaving semantics

- We adopt an asynchronous model.
- At every moment only one entity can be active.
- The entities are the objects, the environment, and the time (for continuous evolution).
- Only one method can be active at a time.
- The active entity determines the set of variables that can change.
  - environment: any variable.
  - time: only continuous variable.
  - method with state machine: only variable affected by transition’s effect.
  - method without state machine: any variable (most abstract assumption).
Summary and discussion

- Language trades off
  1. expressiveness with complexity of analysis.
  2. expressiveness with usability.
- Built on standard temporal logics.
- Tailor-made for ETCS.
- Research directions on hybrid models and on formal analysis.