Programming in Linear Temporal Logic

Wolfgang Jeltsch

Brandenburgische Technische Universität Cottbus
Cottbus, Germany

Seminar talk at the Institute of Cybernetics
Tallinn, Estonia

February 10, 2011
The Temporal Curry–Howard Correspondence

Categorical Semantics for Restricted LTL and FRP

Hybrid Signals

Functional Reactive Dataflow Programming
The Temporal Curry–Howard Correspondence

Categorical Semantics for Restricted LTL and FRP

Hybrid Signals

Functional Reactive Dataflow Programming
Linear Temporal Logic

- trueness of a proposition depends on time
- times are natural numbers
- propositional logic extended with four new constructs:
  - $\circ \varphi$ will hold at the next time
  - $\square \varphi$ will always hold
  - $\Diamond \varphi$ will eventually hold
  - $\varphi \triangleright \psi$ will hold for some time, and then
    $\psi$ will hold
- for now only $\square$ and $\Diamond$:
  - restricted LTL
  - continuous time also possible
Embedding into predicate logic

- temporal formula $\varphi$ can be translated into predicate logic formula $\langle \varphi \rangle$
- $\langle \varphi \rangle$ may contain a single free variable $t$ that denotes the time
- atomic propositions $p$ correspond to predicates $\hat{p}$ that take a time argument
- translation for propositional logic fragment:
  \[
  \langle p \rangle = \hat{p}(t) \quad \langle \varphi \land \psi \rangle = \langle \varphi \rangle \land \langle \psi \rangle \\
  \langle T \rangle = T \quad \langle \varphi \lor \psi \rangle = \langle \varphi \rangle \lor \langle \psi \rangle \\
  \langle \bot \rangle = \bot \quad \langle \varphi \rightarrow \psi \rangle = \langle \varphi \rangle \rightarrow \langle \psi \rangle 
  \]
- translation for $\Box$ and $\Diamond$:
  \[
  \langle \Box \varphi \rangle = \forall t' \in [t, \infty) . \langle \varphi \rangle[t'/t] \\
  \langle \Diamond \varphi \rangle = \exists t' \in [t, \infty) . \langle \varphi \rangle[t'/t] 
  \]
Restricted LTL as a type system

- type inhabitation depends on time
- simple type system extended with two new type constructors □ and ◊
- temporal type α can be translated into dependent type ⟨α⟩
- ⟨α⟩ may contain a single-free variable t that denotes the time
- translation for □ and ◊:

  \[
  \langle \Box \alpha \rangle = \Pi t' \in [t, \infty) . \langle \alpha \rangle[t'/t] \\
  \langle \Diamond \alpha \rangle = \Sigma t' \in [t, \infty) . \langle \alpha \rangle[t'/t] 
  \]

- concepts from Functional Reactive Programming (FRP):
  - □ behaviors
  - ◊ events
- restricted LTL corresponds to a strongly typed form of FRP
- t denotes start times of behaviors and events
The Temporal Curry–Howard Correspondence

Categorical Semantics for Restricted LTL and FRP

Hybrid Signals

Functional Reactive Dataflow Programming
Basics

- Categorical models should be CCCC's:
  - LTL extends propositional logic
  - FRP extends simply-typed $\lambda$-calculus

- Components of a categorical model:
  - Objects: propositions/types
  - Morphisms: time-independent proofs/functions:

$$f : \alpha \rightarrow \beta \Rightarrow f : \Pi t . \langle \alpha \rangle \rightarrow \langle \beta \rangle$$

- $\Box$ and $\Diamond$ are (endo)functors:

$$\begin{align*}
\Box f & : \Box \alpha \rightarrow \Box \beta \\
\Diamond f & : \Diamond \alpha \rightarrow \Diamond \beta
\end{align*}$$

- Start time consistency is ensured:

$$\begin{align*}
\Box : (\Pi t . \langle \alpha \rangle \rightarrow \langle \beta \rangle) & \rightarrow (\Pi t . \langle \Box \alpha \rangle \rightarrow \langle \Box \beta \rangle) \\
\Diamond : (\Pi t . \langle \alpha \rangle \rightarrow \langle \beta \rangle) & \rightarrow (\Pi t . \langle \Diamond \alpha \rangle \rightarrow \langle \Diamond \beta \rangle)
\end{align*}$$
Operations on behaviors

- □ is a comonad:
  
  \[ \text{head} : \square \alpha \to \alpha \]
  
  \[ \text{tails} : \square \alpha \to \square \square \alpha \]

- □ is a strong cartesian functor:
  
  \[ \text{units} : 1 \to \square 1 \]
  
  \[ \text{zip} : \square \alpha \times \square \beta \to \square (\alpha \times \beta) \]

- □ is not an applicative functor:
  
  - lifting of pure values would have to be possible:
    
    \[ \text{const} : \alpha \to \square \alpha \]
    
    \[ \text{const} : \Pi t . \langle \alpha \rangle \to \Pi t' \in [t, \infty) . \langle \alpha \rangle[t'/t] \]
  
  - would break start time consistency:
    
    \[ \text{const} : \Pi t . \langle \alpha \rangle \to \Pi t' \in [t, \infty) . \langle \alpha \rangle[t'/t] \]

- however, this is possible:
  
  \[ f : 1 \to \alpha \]
  
  \[ \square f \circ \text{units} : 1 \to \square \alpha \]
Operations on events

- △ is a monad:
  
  now : \( \alpha \rightarrow \triangle \alpha \)

  join : \( \triangle \triangle \alpha \rightarrow \triangle \alpha \)

- △ is not a strong monad:
  
  time shifting of values would have to be possible:

  \( \text{shift} : \alpha \times \triangle \beta \rightarrow \triangle (\alpha \times \beta) \)

  would break start time consistency:

  \( \text{shift} : \prod t . \langle \alpha \rangle \times \langle \triangle \beta \rangle \rightarrow \Sigma t' \in [t, \infty) . \langle \alpha \rangle[t'/t] \times \langle \beta \rangle[t'/t] \)

- however, △ is □-strong:

  \( \text{age} : \square \alpha \times \triangle \beta \rightarrow \triangle (\square \alpha \times \beta) \)

- sampling can be derived:

  \( \text{sample} : \square \alpha \times \triangle \beta \rightarrow \triangle (\alpha \times \beta) \)

  \( \text{sample} = \triangle (\text{head} \times \text{id}) \circ \text{age} \)
From S4 to restricted LTL

- until now, we have categorical models for CS4/IS4
- no big surprise:
  - classically, restricted LTL is a specialization of S4
  - intuitionistically, it is too
- classical S4 and restricted LTL differ in their restrictions on the accessibility relation:
  - S4 reflexive order
  - restr. LTL total reflexive order
- add a further operation that ensures totality of time:

  \[
  \text{race} : \Diamond \alpha \times \Diamond \beta \to \Diamond (\alpha \times \beta + \alpha \times \Diamond \beta + \Diamond \alpha \times \beta)
  \]

- possible outcomes of time comparison represented by the different alternatives:

  \[
  = \alpha \times \beta \\
  < \alpha \times \Diamond \beta \\
  > \Diamond \alpha \times \beta
  \]
The Temporal Curry–Howard Correspondence

Categorical Semantics for Restricted LTL and FRP

Hybrid Signals

Functional Reactive Dataflow Programming
-LTL and its corresponding FRP dialect

- translation of ▷-formulas into predicate logic formulas:
  \[ \langle \varphi \ ▷ \psi \rangle = \exists t' \in (t, \infty) . (\forall t'' \in [t, t') . \langle \varphi \rangle[t''/t]) \land \langle \psi \rangle[t'/t] \]

- ▷ as a type constructor of FRP:
  \[ \langle \alpha \ ▷ \beta \rangle = \Sigma t' \in (t, \infty) . (\Pi t'' \in [t, t') . \langle \alpha \rangle[t''/t]) \times \langle \beta \rangle[t'/t] \]

- components of a value of type \( \alpha \ ▷ \beta \):
  - a finite behavior with values of type \( \alpha \)
  - a terminating event with a value of type \( \beta \)

- introduction of weak variant of ▷ that does not guarantee termination

- notation:
  - ▷⊥ strong variant (▷ as defined above)
  - ▷⊤ weak variant

- □ and ◇ now derivable:
  - □\( \alpha \) = \( \alpha \ ▷ \top \) 0
  - ◇\( \beta \) = \( \beta \ + \ 1 \ ▷ \bot \)
Applications of ▶-types

▶ ▶-types are useful as such:
  ▶ temperatures from some sensor that may be detached from the computer:

  \[ \mathbb{R} \triangleright_{\top} 1 \]

  ▶ dialog window:

  \[ \text{UI} \triangleright_{\top} \alpha \]

  etc.

▶ ▶-types are useful in combination with (co)induction:
  ▶ audio signal that may switch between stereo and mono:

  \[ \nu \sigma . (\mathbb{R} \times \mathbb{R}) \triangleright_{\top} \mathbb{R} \triangleright_{\top} \sigma \]

  ▶ positions of a pen that might be taken off from the drawing area:

  \[ \nu \sigma . (\mathbb{R} \times \mathbb{R}) \triangleright_{\top} 1 \triangleright_{\top} \sigma \]

  etc.
The ▷-functor

- categorical model \( C \) is a CCCC
- derive a category \( U \) from \( C \):

\[
\text{Obj } U = \text{Obj } C \times \text{Obj } C \times \{\bot, \top\}
\]

\[
\text{hom}((\alpha_1, \beta_1, w_1), (\alpha_2, \beta_2, w_2)) =
\begin{cases} 
\text{hom}(\alpha_1, \alpha_2) \times \text{hom}(\beta_1, \beta_2) & \text{if } w_1 \leq w_2 \\
\emptyset & \text{otherwise}
\end{cases}
\]

- ▷ is a functor from \( U \) to \( C \)
- notation:

\[
\alpha \triangleright_w \beta = \triangleright(\alpha, \beta, w)
\]

- applying ▷ to morphisms allows for several things:
  - mapping of values of the behavior part
  - mapping of value of the terminating event
  - weakening
Comonadic and monadic structure

- \( \_ \triangleright_w \beta \) is a comonad:
  
  \[
  \text{head} : \alpha \triangleright_w \beta \to \alpha \\
  \text{tails} : \alpha \triangleright_w \beta \to (\alpha \triangleright_w \beta) \triangleright_w \beta
  \]

- \( \beta = 0 \) and \( w = \top \) leads to comonadic structure of \( \Box \)

- \( \alpha \triangleright_w \_ \) is an ideal monad:
  
  \[
  \text{optjoin} : \alpha \triangleright_w (\beta + \alpha \triangleright_w \beta) \to \alpha \triangleright_w \beta
  \]

- monad can be derived:
  
  \[
  \begin{align*}
  \text{now} : & \quad \beta \to (\beta + \alpha \triangleright_w \beta) \\
  \text{join} : & \quad (\beta + \alpha \triangleright_w \beta) + \alpha \triangleright_w (\beta + \alpha \triangleright_w \beta) \to \beta + \alpha \triangleright_w \beta
  \end{align*}
  \]

- \( \alpha = 1 \) and \( w = \bot \) leads to monadic structure of \( \Diamond \)
Monoidal structure

- make $U$ a symmetric monoidal category:

$$(\alpha_1, \beta_1, w_1) \otimes (\alpha_2, \beta_2, w_2) = (\alpha_1 \times \alpha_2, \rho, w_1 \sqcap w_2)$$

$I = (1, 0, \top)$

where

$$\rho = \beta_1 \times \beta_2 + \beta_1 \times \alpha_2 \triangleright w_2 \beta_2 + \alpha_1 \triangleright w_1 \beta_1 \times \beta_2$$

- $\triangleright$ is a strong symmetric monoidal functor from $U$ to $C$:

merge : $\alpha_1 \triangleright w_1 \beta_1 \times \alpha_2 \triangleright w_2 \beta_2 \rightarrow \alpha_1 \times \alpha_2 \triangleright w_1 \sqcap w_2 \rho$

never : $1 \triangleright_{\top} 0$
Specializations

- ▶ is a strong symmetric monoidal functor from $U$ to $C$:
  
  $\text{merge} : \alpha_1 \triangleright_{w_1} \beta_1 \times \alpha_2 \triangleright_{w_2} \beta_2 \rightarrow \alpha_1 \times \alpha_2 \triangleright_{w_1 \cap w_2} \rho$

  $\text{never} : 1 \triangleright_T 0$

  where

  $\rho = \beta_1 \times \beta_2 + \beta_1 \times \alpha_2 \triangleright_{w_2} \beta_2 + \alpha_1 \triangleright_{w_1} \beta_1 \times \beta_2$

- ▶ strong cartesian functor structure of $\Box$:

  $\beta_1 = \beta_2 = 0 \quad w_1 = w_2 = T$

- ▶ from merge to age:

  $\beta_1 = 0 \quad w_1 = T$

  $\alpha_2 = 1 \quad w_2 = \bot$

- ▶ from merge to race:

  $\alpha_1 = \alpha_2 = 1 \quad w_1 = w_2 = \bot$
The inverse of merge

- the type of the terminating event:

\[ \rho = \beta_1 \times \beta_2 + \beta_1 \times \alpha_2 \triangleright_{w_2} \beta_2 + \alpha_1 \triangleright_{w_1} \beta_1 \times \beta_2 \]

- drop information from the terminating event:

\[ \text{restrict}_i : \rho \rightarrow \beta_i + \alpha_i \triangleright_{w_i} \beta_i \]
\[ \text{restrict}_i = [\iota_1 \circ \pi_i, \iota_i \circ \pi_i, \iota_{1-i} \circ \pi_i] \]

- recover the original \( \triangleright \)-values:

\[ \text{recover}_i : \alpha_1 \times \alpha_2 \triangleright_{w_1 \cap w_2} \rho \rightarrow \alpha_i \triangleright_{w_i} \beta_i \]
\[ \text{recover}_i = \text{optjoin} \circ (\pi_i \triangleright \text{restrict}_i) \]

- combine the recovered values:

\[ \text{merge}^{-1} : \alpha_1 \times \alpha_2 \triangleright_{w_1 \cap w_2} \rho \rightarrow \alpha_1 \triangleright \beta_1 \times \alpha_2 \triangleright \beta_2 \]
\[ \text{merge}^{-1} = \langle \text{recover}_1, \text{recover}_2 \rangle \]
The Temporal Curry–Howard Correspondence

Categorical Semantics for Restricted LTL and FRP

Hybrid Signals

Functional Reactive Dataflow Programming
in LTL and FRP

- use \( \mathbb{N} \) as the set of times
- translation of \( \Diamond \)-formulas into predicate logic formulas:
  \[
  \langle \Diamond \varphi \rangle = \langle \varphi \rangle[t + 1/t]
  \]
- \( \Diamond \) as a type constructor of FRP:
  \[
  \langle \Diamond \alpha \rangle = \langle \alpha \rangle[t + 1/t]
  \]
- value of type \( \Diamond \alpha \) is a value of type \( \alpha \) occurring at the next time
- semantically, \( \Diamond \) is just a strong cartesian functor:
  \[
  \begin{align*}
  f : \alpha &\to \beta \\
  \Diamond f : \Diamond \alpha &\to \Diamond \beta \\
  \text{unit} : 1 &\to \Diamond 1 \\
  \text{pair} : \Diamond \alpha \times \Diamond \beta &\to \Diamond (\alpha \times \beta)
  \end{align*}
  \]
Deriving the other constructs

- □, ◊, and ▶ derivable via induction and coinduction:

  □α = νσ . α × ◯σ
  ◊β = µσ . β + ◯σ
  α ▶⊥ β = µσ . α × ◯(β + σ)
  α ▶⊤ β = νσ . α × ◯(β + σ)

- interesting exercise:
  - derive all operations of ▶-FRP from the ◯-operations
  - proof that the derived operations fulfill the necessary laws
Advanced dataflow programming

- FRP is a kind of dataflow language:
  - streams over $\alpha$:
    \[ \Box \alpha \]
  - partial streams over $\alpha$:
    \[ (1 + \alpha) \times \nu \sigma . \ 1 \triangleright \top (\alpha \times \sigma) \]
- more powerful than traditional dataflow languages:
  - productive partial streams over $\alpha$:
    \[ (1 + \alpha) \times \nu \sigma . \ 1 \triangleright \bot (\alpha \times \sigma) \]
  - streams with values of different type
Shifting

- fby operator appends a stream to an initial value:
  \[ \text{fby} : \alpha \times \Box \alpha \rightarrow \Box \alpha \]

- needs to shift values to the future
- cannot be done implicitly, since it would break start time consistency
- can be made possible by introducing tensorial strength:
  \[ \text{shift} : \alpha \times \Diamond \beta \rightarrow \Diamond (\alpha \times \beta) \]

- simpler operator is sufficient:
  \[ \text{later} : \alpha \rightarrow \Diamond \alpha \]

- \(\Diamond\) is now an applicative functor
Programming in Linear Temporal Logic

Wolfgang Jeltsch

Brandenburgische Technische Universität Cottbus
Cottbus, Germany

Seminar talk at the Institute of Cybernetics
Tallinn, Estonia

February 10, 2011