Evaluation Tree languages

Igor Yanovich

MIT

July 27, 2010

Logic Colloquium 2010, Paris
Main points

- Evaluation Tree ($ET$) semantics allows for formulation of more fine-grained systems extending Modal Logic ($ML$) invariant up to generated submodels.

- A semi-syntactic, semi-semantic notion of expressivity at the level of subformulas: focusing on the range of context-sensitive operators definable in a logic.
A more detailed plan

1. Quickly review of standard modal logic (ML), a couple of hybrid logics (HL), and classical expressivity for them

2. Motivate non-classical “operator expressivity”, or subformula expressivity, by way of examples

3. Introduce Evaluation tree (ET) languages

4. Show that even classically very mild ET systems may be more subformula-expressive than more classically powerful hybrid systems
Standard modal logic (ML)

- **ML** is invariant up to (van Benthem’s) bisimulations.
- Bisimulation is much weaker than isomorphism:

\[
\begin{array}{c}
m_2 : p, q \\
m_1 : p \\
m_3 : p, q \\
m_4 : q \\
\end{array}
\]

\[
\begin{array}{c}
m_2 : p, q \\
m_1 : p \\
m_3 : p, q \\
m_4 : q \\
\end{array} ightleftharpoons
\begin{array}{c}
n_1 : p \\
n_2 : p, q \\
n_3 : q \\
\hfill \emptyset
\end{array}
\]

\[
Z = \{ \langle m_1, n_1 \rangle, \langle m_2, n_2 \rangle, \langle m_3, n_2 \rangle, \langle m_4, n_3 \rangle \} \Rightarrow \text{bisimulation}
\]

- \( xZy \Rightarrow x \text{ and } y \text{ are modally equivalent} \)
Making **ML** more expressive: Hybrid logic (HL)

- Need more expressivity, but less than the full FOL?

- Hybrid languages! ([Blackburn and Seligman, 1995])

- **Nominals**: propositional variables true at exactly one point (=names for points)

- **Nominal variables** $x, y, \ldots$: nominals open for binding

- **Hybrid semantics** is like Kripke semantics, but with an additional evaluation parameter: an *assignment function* storing the nominal variables bound.

\[ M, w, g \models \phi \]
In particular, we are interested in *H*-*L* logics invariant *up to generated submodels*

[Areces et al., 1999]: *H*L(↓, @) precisely corresponds to the fragment of FOL invariant over generated submodels (*FOL*Bou, the bounded fragment of FOL):

\[ \text{H*L}(\downarrow, @) \leq \text{FOL*Bou}, \quad \text{FOL*Bou} \leq \text{H*L}(\downarrow, @) \]
HL for generated submodels: $\text{HL}(\downarrow, @)$

- $\downarrow$ stores a point in the assignment function for future use
  \[
  M, w, g \models \downarrow x. \phi \iff M, w, g^{[x \rightarrow w]} \models \phi
  \]
  
  Example: $\downarrow x. \lozenge x$ is equivalent to $Rww$
  $\Rightarrow$ unlike $\text{ML}$, $\text{HL}(\downarrow)$ can express $R$-reflexivity

- $@$ shifts the current evaluation point to a previously stored one
  \[
  M, w, g \models \!@x. \phi \iff M, g(x), g \models \phi
  \]
  
  Example: $\downarrow x. \lozenge (\downarrow y. @x. \lozenge \neg y)$ is equivalent to $\exists_2 v : Rvw$
  $\Rightarrow$ unlike $\text{HL}(\downarrow)$, $\text{HL}(\downarrow, @)$ can do counting

$\text{ML} < \text{HL}(\downarrow) < \text{HL}(\downarrow, @)$
Why look at expressivity for subformulas?

Some examples:

- **Task 1**: Your modal formula looks into a transition system, and when you encounter a certain operator in the formula, no matter at which level of embedding, you need to check whether you saw some other symbol up to three transitions from the beginning of evaluation.

- **Task 2**: A certain operator signals that you should go exactly two steps of evaluation back, and check if some $p$ holds there.

- **Task 3**: Your formula looks into a database (a relational structure), and at some embedded level you want to check whether some object $x$ had a property $Q$ at the point(s) where your formula encountered $x$ for the first time.
Why look at expressivity for subformulas

- **HL**($\downarrow, \ominus$) can express a *specific* checking procedure for each particular situation where the task may arise.

- But **HL**($\downarrow, \ominus$) cannot express a *function* from situations to outcomes of the task.

- There is a natural sense in which systems which allow to express queries like those uniformly are more expressive than systems which cannot.

- **Our next step**: we explore subformula-expressivity in the domain bounded by invariance over generated submodels.
Evaluation Tree semantics for standard ML

- **HL** adds an *assignment function* parameter
- **ET** adds an *evaluation tree* parameter

We use brute force: we store a huge chunk of the model, all points which were used while checking the formula up to the current point.

Formulas are evaluated at a Kripke model \( M \) and an evaluation tree \( \mathcal{F} \): a rooted tree labeled with points of the structure.

One of the nodes of \( \mathcal{F} \) is designated as the current vertex \( \text{cur}(\mathcal{F}) \). For visual similarity to **ML**, we use the current node as a third parameter.

Some notation:

- \( ^* \nu \), \( \nu \) a vertex in \( \mathcal{F} \), is the point in \( M \) that \( \nu \) is labeled with
- \( \mathcal{F} +_w Z \), \( Z \) a set, \( w \) a vertex in \( \mathcal{F} \), adds all members of \( Z \) as daughters to \( w \) in \( \mathcal{F} \)
- \( dtr(\mathcal{F}, \nu) \) is the set of all daughter vertices of \( \nu \) in \( \mathcal{F} \)
Evaluation Tree languages

Evaluation Tree semantics for standard **ML**

- **Truth at a tree:**
  - \( M, \mathcal{F}, w \models_{ET} p \) iff \( w \in V(p) \)
  - \( M, \mathcal{F}, w \models_{ET} \bot \) never
  - \( M, \mathcal{F}, w \models_{ET} \neg \phi \) iff \( M, \mathcal{F}, w \not\models \phi \)
  - \( M, \mathcal{F}, w \models_{ET} \phi \land \psi \) iff \( M, \mathcal{F}, w \models \phi \) and \( M, \mathcal{F}, w \models \psi \)
  - \( M, \mathcal{F}, w \models_{ET} \diamond \phi \) iff for some \( v \in dtr(\mathcal{F} + w \{v | R(*w)(*v)\}, w) \),
    \[
    M, (\mathcal{F} + w \{v | R(*w)(*v)\}), v \models \phi
    \]

- Let \( T_w \) be a single-vertex tree with \( w \) the only node. Then, for an **ML** formula \( \phi \):
  - \( M, T_w, w \models_{ML} \phi \) iff \( M, *w \models_{ET} \phi \)
Evaluation Tree operators

- The evaluation point parameter $w$ cannot store too much information; the evaluation tree parameter $\mathcal{F}$ stores a lot more, and we can use it.

- Family of $\{.,.\}$ operators:
  - First argument $g : \text{eval. trees} \mapsto \text{sets of nodes}$
  - Second argument $\phi$: just an ET formula

  $$\mathcal{M}, \mathcal{F} \models \{g, \phi\} \iff \text{set } g(\mathcal{F}) \text{ is non-empty, and for each } \ast \nu \in g(\mathcal{F}), \mathcal{M}, \nu \models \phi$$

- By controlling the language of function descriptions $g$, we control the expressivity
Evaluation Tree operators

Examples of function descriptions:

- \(\text{root}(F)\) returns the root node of \(F\)
- \(g \downarrow (F)\) returns the union of the daughter sets of \(v \in g(F)\)
- \(3(F)\) returns \(\text{root}(F) \cup \text{root}(g(F)) \cup \text{root}(\downarrow(F))\)
- \(\uparrow(F)\) returns the parent of \(\text{cur}(F)\) together with its sisters
- \(\uparrow_2(F)\) returns the grandparent of \(\text{cur}(F)\) together with its sisters
- \(\rhd^2(F)\) if the parent has only one sister, returns the parent and the sister; returns empty set otherwise

- \(\{3, .\}\) performs Task 1.
- \(\{\uparrow_2, .\}\) performs Task 2.
Evaluation Tree operators

For Task 3 (finding out whether \( x \) had \( Q \) the first time we saw it), we need more than \{.,.\}, and modal predicate logic as the basis for our ET system.

So we only illustrate informally how that would work:

operator \( ?_{1}^{\psi(y)}x \), \( x \) an individual, \( \psi(y) \) a formula in one variable \( y \), goes up to \( \text{root}(F) \), and starts going down from there until it sees \( x \) for the first time. When \( x \) is found, it is checked whether \( \psi(x) \) holds at that point.

⇒ unsurprisingly, we can formulate many sorts of operators using the extra information we store in the evaluation tree
Translations from \textbf{ET} into \textbf{ML} and \textbf{HL}($\downarrow$, $\varnothing$)

- But all the operators we have shown so far do not make the logic more expressive at the level of formulas: given a \textbf{ET} formula featuring them, one can always produce an equivalent standard formula\(^1\)

- For example:
  \[
  \begin{align*}
    \Diamond \Diamond \{\text{root}\&, p\} & \equiv \Diamond \Diamond T \land \Box p \\
    \Diamond \Diamond \Diamond \{\text{root}\&, p\} & \equiv \Diamond \Diamond \Diamond T \land \Box p \\
    \Diamond \Diamond \{\uparrow_2, p\} & \equiv \Diamond \Diamond T \land p \\
    \Diamond \Diamond \Diamond \{\uparrow_2, p\} & \equiv \Diamond \Diamond \Diamond T \land \Box p
  \end{align*}
  \]

- It is easy to show that any \textbf{ET} with just \{..\} being a non-standard primitive is invariant over generated submodels, so \textbf{HL}($\downarrow$, $\varnothing$) is the upper bound for classical expressivity of \textbf{ET}\{\}

---

\(^1\)For a classically more expressive \textbf{ET} operator, think of one counting \{v \mid wRv\}. 
Yet even very mild ET systems can express operators which even classically stronger HL(\downarrow, @) cannot express. Once again:

\[
\diamond \diamond \{ \text{root} \downarrow, p \} \equiv \diamond \diamond \text{T} \land \Box p \\
\diamond \diamond \diamond \{ \text{root} \downarrow, p \} \equiv \diamond \diamond \diamond \text{T} \land \Box p
\]

So in what follows we work with mildest ET systems: even their power can show the point.

A side remark: We can view \{., .\} operators as rewriting instructions for generating ML and HL(\downarrow, @) formulas. \{\} expressions have constant meanings in terms of a rewriting algorithm for ML/HL(\downarrow, @), but do not have a single ML/HL(\downarrow, @) formula expressing their meaning.
Expressivity at subformulas

- Modal language $L_1$ is not more expressive than modal language $L_2$, $L_1 \leq_{sf} L_2$, iff there is a translation function $Tr$ from formulas of $L_1$ into formulas of $L_2$ such that for every $M$, $w$, formulas $\phi(p), \psi$ of $L_1$,

$$M, w \models_{L_1} \phi^\psi_p \iff M, w \models_{L_2} Tr(\phi) \frac{Tr(\psi)}{Tr(p)},$$

where $\phi^\psi_p$ is $\phi$ with $\psi$ substituted for $p$.

- $\leq_f$ is the standard notion of “not more expressive than”, at the level of formulas
Expressivity at subformulas

• Obviously, \((L_1 =_{sf} L_2) \Rightarrow (L_1 =_f L_2)\)

• But in general, \(\leq_{sf}\) is **not** a refinement of \(\leq_f\).

In particular, there exist \(sf\)-incomparable languages which are \(f\)-equivalent:

Consider \(ET\, root\) and \(ET\, \uparrow\): two logics with a single \(ET\) operator each.

\[ ET\, root =_f ML =_f ET\, \uparrow. \]

But a subformula \(\{root, p\}\) is not expressible in \(ET\, \uparrow\):

\[ \Diamond\{root, p\} = \Diamond\{\uparrow, p\}, \quad \text{and} \quad \Diamond\Diamond\{root, p\} = \Diamond(\{\uparrow, p\} \land \Diamond T) \]

And vice versa, \(\{\uparrow, p\}\) is not expressible in \(ET\, root\):

\[ \Diamond\{\uparrow, p\} = \Diamond\{root, p\}, \quad \text{while} \quad \Diamond\Diamond\{\uparrow, p\} = \Diamond(\{\varnothing, p\} \land \Diamond T). \]
In fact, things are even worse: even with $L_1 <_f L_2$, $L_1$ and $L_2$ may be $sf$-incomparable.

Consider $ET_{root}$ and $HL(\downarrow)$.

Since $ET_{root} \leq_f ML$, and $ML$ is a fragment of $HL(\downarrow)$, we have $ET_{root} \leq_f HL(\downarrow)$.

On the other hand, $\downarrow x. \unlhd x \in HL(\downarrow)$ expresses reflexivity, and $ET_{root}$ cannot express reflexivity, lacking nominals. So $HL(\downarrow) \nleq_f ET_{root}$, and thus $ET_{root} <_f HL(\downarrow)$, and $HL(\downarrow) \nleq_{sf} ET_{root}$.

But at the same time, $HL(\downarrow)$, without $\circ$, cannot shift interpretation from an embedded context to a higher context. So it is not possible to express even the simple $\{root, p\}$ as a $HL(\downarrow)$ operator, hence $ET_{root} \nleq_{sf} HL(\downarrow)$.

Thus $ET_{root} <_f HL(\downarrow)$, but they are $\leq_{sf}$-incomparable.
What subformula expressivity means

- $f$-expressivity is about statements you can make
- $sf$-expressivity is about context-sensitive, within-formula statements

- When $L_1 <_{sf} L_2$, $L_2$ has more ways to make context-sensitive statements

- When $L_1$ and $L_2$ are $sf$-incomparable, it means that either has at least one powerful device for making context-sensitive statements, but those devices allow different, mutually inexpressible kinds of context-sensitivity.
Some possible future directions

- **sf**-expressivity:
  - Study **sf**-expressivity relations, drawing the $<_\text{sf}$-map of the expressivity region of invariance over generated submodels.
  - Study the relations between $f$- and **sf**-expressivity (e.g., under which conditions $<_f$ implies $<_\text{sf}$).
  - Check whether the notion of **sf**-expressivity has natural uses outside of the generated submodels region.

- Evaluation tree languages:
  - Study the structure of the space of **ET** operators.
  - Check the proof-theoretic side of things for **ET**, including the relations between **ET** and tableaux methods for classical **ML** and richer systems.
  - [Blackburn and Seligman, 1998] suggest studying indexicality logics by embedding them into **HL**; embedding into **ET** is a better choice.
References

Hybrid logic is the bounded fragment of first order logic.

Hybrid languages.

What are hybrid languages?

The author is particularly grateful for the discussions with, and the comments of Cameron Freer, Salvador Mascarenhas, and Mike Solomon, at MIT and NYU.
All errors, are, of course, my responsibility only.