Incremental Theme Verbs

- Certain NP's measure out the event. They are direct objects consumed or created in increments over time (cf. *eat an apple* vs. *push a chart*) (Tenny 1994).

Incremental Theme Verbs

- Certain NP's measure out the event. They are direct objects consumed or created in increments over time (cf. *eat an apple* vs. *push a chart*) (Tenny 1994).
- In *Mary drank a glass of wine* "every part of the glass of wine being drunk corresponds to a part of the drinking event" (Krifka 1992)

Incremental Theme Verbs

- Certain NP's measure out the event. They are direct objects consumed or created in increments over time (cf. *eat an apple* vs. *push a chart*) (Tenny 1994).
- In *Mary drank a glass of wine* "every part of the glass of wine being drunk corresponds to a part of the drinking event" (Krifka 1992)
- "Incremental themes are arguments that are completely processed only upon termination of the event, i.e., at its end point" (Dowty 1991).

Degree Achievements

- Verbs with variable aspectual behavior: they seems to be change of state verbs like other achievements , but allow durational adverbs (Dowty 1979, Hay, Kennedy and Levin 1999, Rappaport Hovav 2008).

Degree Achievements

- Verbs with variable aspectual behavior: they seems to be change of state verbs like other achievements , but allow durational adverbs (Dowty 1979, Hay, Kennedy and Levin 1999, Rappaport Hovav 2008).
- No implication that exactly the same change of state took place over and over again (no semelfactives).

Degree Achievements

- Verbs with variable aspectual behavior: they seems to be change of state verbs like other achievements , but allow durational adverbs (Dowty 1979, Hay, Kennedy and Levin 1999, Rappaport Hovav 2008).
- No implication that exactly the same change of state took place over and over again (no semelfactives).
- Scalar predicates: verbs which lexically specify a change along a scale inasmuch as they denote an ordered set of values for a property of an event argument (Hay, Kennedy and Levin 1999, Rappaport Hovav 2008).

Degree Achievements

- Verbs with variable aspectual behavior: they seems to be change of state verbs like other achievements , but allow durational adverbs (Dowty 1979, Hay, Kennedy and Levin 1999, Rappaport Hovav 2008).
- No implication that exactly the same change of state took place over and over again (no semelfactives).
- Scalar predicates: verbs which lexically specify a change along a scale inasmuch as they denote an ordered set of values for a property of an event argument (Hay, Kennedy and Levin 1999, Rappaport Hovav 2008).
- For example *cool*, *age*, *lenghten*, *shorten*; *descend*.
- *Let the soup cool for 10 minutes*.
- *I went on working until the soup cooled*.

Subatomic Event Structure

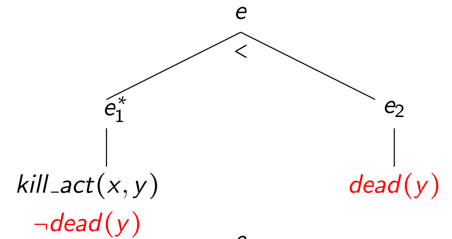
Pustejovsky (1991)

(17)
a. EVENT → STATE | PROCESS | TRANSITION
b. STATE: → e
c. PROCESS: → $e_1 \ldots e_n$
d. TRANSITION_{ach}: → STATE STATE
e. TRANSITION_{acc}: → PROCESS STATE

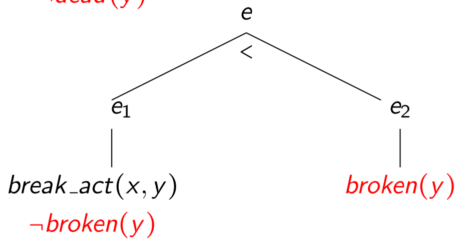
Opposition Structure

Pustejovsky (2000)

(18) kill



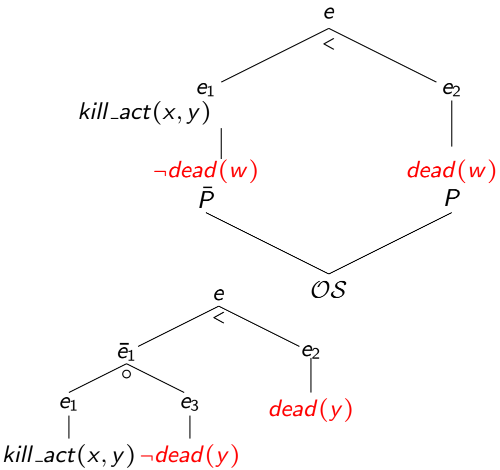
(19) break



Qualia Structure with Opposition Structure

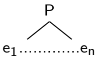
$$\begin{bmatrix} \textbf{kill} & & \\ \text{EVENTSTR} & = & \begin{bmatrix} \text{E}_0 = \textbf{e}_0\textbf{:state} \\ \text{E}_1 = \textbf{e}_1\textbf{:process} \\ \text{E}_2 = \textbf{e}_2\textbf{:state} \\ \text{RESTR} = <_{\propto} \\ \text{HEAD} = \textbf{e}_1 \end{bmatrix} \end{bmatrix}$$

Opposition is Part of Event Structure

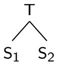


(20) a. STATE: a simple event, evaluated without referring to other events: *be sick*, *love*, *know*

b. PROCESS: a sequence of events identifying the same semantic expression: *run*, *push*, *drag*



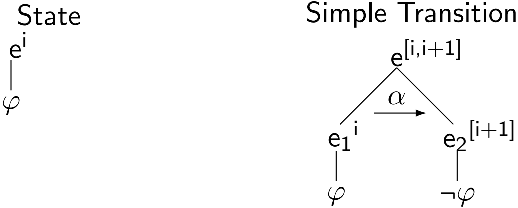
c. TRANSITION: an event identifying a semantic expression evaluated with respect to its opposition: *give*, *open*; *build*:
Binary transition (achievement): $\neg\phi \in S_1$, and $\phi \in S_2$



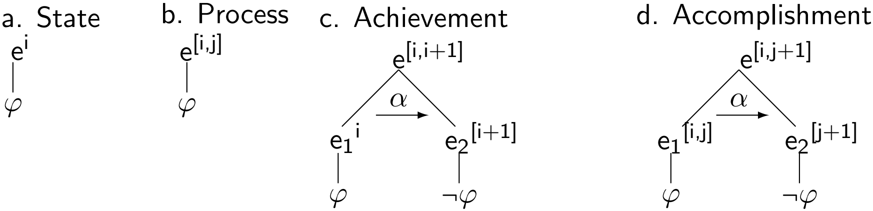
Complex transition (accomplishment): $\neg\phi \in P$, and $\phi \in S$

Dynamic Event Models (Pustejovsky, 2013)

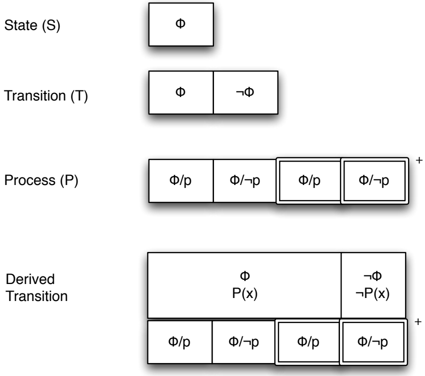
Two Primitive Event Types



Derived Vendler Event Types



Frame-based Event Structure



2nd Conference on CTF, Pustejovsky (2009)

Dynamic Event Model

Pustejovsky and Moszkowicz (2011), Pustejovsky (2013)

- Events are built up from multiple (stacked) layers of primitive constraints on the individual participants.
- There may be many changes taking place within one atomic event, when viewed at the subatomic level.

- Formulas: ϕ propositions. Evaluated in a state, s.

- Formulas: ϕ propositions. Evaluated in a state, s.
- Programs: α, functions from states to states, $s \times s$. Evaluated over a pair of states, (s, s').

- Formulas: ϕ propositions. Evaluated in a state, s.
- Programs: α, functions from states to states, $s \times s$. Evaluated over a pair of states, (s, s').
- Temporal Operators: $\bigcirc\phi$, $\Diamond\phi$, $\Box\phi$, $\phi\mathcal{U}\psi$.

Dynamic Event Model

Dynamic Interval Temporal Logic

- Formulas: ϕ propositions. Evaluated in a state, s.
- Programs: α, functions from states to states, $s \times s$. Evaluated over a pair of states, (s, s').
- Temporal Operators: $\bigcirc\phi$, $\Diamond\phi$, $\Box\phi$, $\phi\mathcal{U}\psi$.
- Program composition:

Dynamic Event Model

Dynamic Interval Temporal Logic

- Formulas: ϕ propositions. Evaluated in a state, s.
- Programs: α, functions from states to states, $s \times s$. Evaluated over a pair of states, (s, s').
- Temporal Operators: $\bigcirc\phi$, $\Diamond\phi$, $\Box\phi$, $\phi\mathcal{U}\psi$.
- Program composition:
  1. They can be ordered, $\alpha;\beta$ (α is followed by β);

- Formulas: ϕ propositions. Evaluated in a state, s.
- Programs: α, functions from states to states, $s \times s$. Evaluated over a pair of states, (s, s').
- Temporal Operators: $\bigcirc\phi$, $\Diamond\phi$, $\Box\phi$, $\phi\mathcal{U}\psi$.
- Program composition:
  1. They can be ordered, $\alpha;\beta$ (α is followed by β);
  2. They can be iterated, a^* (apply a zero or more times);

- Formulas: ϕ propositions. Evaluated in a state, s.
- Programs: α, functions from states to states, $s \times s$. Evaluated over a pair of states, (s, s').
- Temporal Operators: $\bigcirc\phi$, $\Diamond\phi$, $\Box\phi$, $\phi\mathcal{U}\psi$.
- Program composition:
  1. They can be ordered, $\alpha;\beta$ (α is followed by β);
  2. They can be iterated, a^* (apply a zero or more times);
  3. They can be disjoined, $\alpha \cup \beta$ (apply either α or β);

- Formulas: ϕ propositions. Evaluated in a state, s.
- Programs: α, functions from states to states, $s \times s$. Evaluated over a pair of states, (s, s').
- Temporal Operators: $\bigcirc\phi$, $\Diamond\phi$, $\Box\phi$, $\phi\mathcal{U}\psi$.
- Program composition:
  1. They can be ordered, $\alpha;\beta$ (α is followed by β);
  2. They can be iterated, a^* (apply a zero or more times);
  3. They can be disjoined, $\alpha \cup \beta$ (apply either α or β);
  4. They can be turned into formulas
     $[\alpha]\phi$ (after every execution of α, ϕ is true);
     $\langle\alpha\rangle\phi$ (there is an execution of α, such that ϕ is true);

- Formulas: ϕ propositions. Evaluated in a state, s.
- Programs: α, functions from states to states, $s \times s$. Evaluated over a pair of states, (s, s').
- Temporal Operators: $\bigcirc\phi$, $\Diamond\phi$, $\Box\phi$, $\phi\mathcal{U}\psi$.
- Program composition:
  1. They can be ordered, $\alpha;\beta$ (α is followed by β);
  2. They can be iterated, a^* (apply a zero or more times);
  3. They can be disjoined, $\alpha \cup \beta$ (apply either α or β);
  4. They can be turned into formulas
     $[\alpha]\phi$ (after every execution of α, ϕ is true);
     $\langle\alpha\rangle\phi$ (there is an execution of α, such that ϕ is true);
  5. Formulas can become programs, $\phi?$ (test to see if ϕ is true, and proceed if so).

Dynamic Event Model

(21) a. Mary was sick today.
b. My phone was expensive.
c. Sam lives in Boston.

Dynamic Event Model

(22) a. Mary was sick today.
b. My phone was expensive.
c. Sam lives in Boston.

We assume that a *state* is defined as a single frame structure (event), containing a proposition, where the frame is temporally indexed, i.e., $e^i \rightarrow \phi$ is interpreted as ϕ holding as true at time i. The frame-based representation from Pustejovsky and Moszkowicz (2011) can be given as follows:

Dynamic Event Model

(23) $\boxed{\phi}_{e}^{i}$

Dynamic Event Model

(26) $\boxed{\phi}_e^i$

Propositions can be evaluated over subsequent states, of course, so we need an operation of concatenation, +, which applies to two or more event frames, as illustrated below.

Dynamic Event Model

(29) $\boxed{\phi}_e^i$

Propositions can be evaluated over subsequent states, of course, so we need an operation of concatenation, +, which applies to two or more event frames, as illustrated below.

(30) $\boxed{\phi}_e^i + \boxed{\phi}_e^j = \boxed{\phi}_e^{[i,j]}$

(32) $\boxed{\phi}^{i}_{e}$

Propositions can be evaluated over subsequent states, of course, so we need an operation of concatenation, $+$, which applies to two or more event frames, as illustrated below.

(33) $\boxed{\phi}^{i}_{e} + \boxed{\phi}^{j}_{e} = \boxed{\phi}^{[i,j]}_{e}$

Semantic interpretations for these are:

Dynamic Event Model

(35) $\boxed{\phi}_e^i$

Propositions can be evaluated over subsequent states, of course, so we need an operation of concatenation, +, which applies to two or more event frames, as illustrated below.

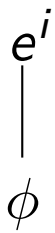
(36) $\boxed{\phi}_e^i + \boxed{\phi}_e^j = \boxed{\phi}_e^{[i,j]}$

Semantic interpretations for these are:

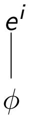
(37) a. $[[\boxed{\phi}]]_{\mathbf{M},i} = 1 \text{ iff } V_{\mathbf{M},i}(\phi) = 1.$

b. $[[\boxed{\phi}\boxed{\phi}]]_{\mathbf{M},\langle i,j\rangle} = 1 \text{ iff } V_{\mathbf{M},}(\phi) = 1 \text{ and } V_{\mathbf{M},j}(\phi) = 1,$ where $i < j$.

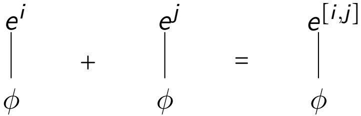
Dynamic Event Model



Dynamic Event Model



Tree structure for event concatenation:



An LTS consists of a 3-tuple, $\langle S, Act, \rightarrow \rangle$, where

An LTS consists of a 3-tuple, $\langle S, Act, \rightarrow \rangle$, where

(40) a. S is the set of states;
b. Act is a set of actions;
c. $\rightarrow$ is a total transition relation: $\rightarrow \subseteq S \times Act \times S$.

Dynamic Event Model

Labeled Transition System (LTS)

An LTS consists of a 3-tuple, $\langle S, Act, \rightarrow \rangle$, where

(42) a. S is the set of states;
b. Act is a set of actions;
c. $\rightarrow$ is a total transition relation: $\rightarrow \subseteq S \times Act \times S$.

(43) $(e_1, \alpha, e_2) \in \rightarrow$

(cf. also Fernando (2001, 2013)

An action, α provides the labeling on an arrow, making it explicit what brings about a state-to-state transition.

An action, α provides the labeling on an arrow, making it explicit what brings about a state-to-state transition.

As a shorthand for

Dynamic Event Model

Labeled Transition System (LTS)

An action, α provides the labeling on an arrow, making it explicit what brings about a state-to-state transition.

As a shorthand for

(46) a. $(e_1, \alpha, e_2) \in \rightarrow$, we will also use:

An action, α provides the labeling on an arrow, making it explicit what brings about a state-to-state transition.

As a shorthand for

(47) a. $(e_1, \alpha, e_2) \in \rightarrow$, we will also use:

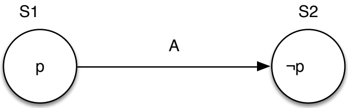
b. $e_1 \xrightarrow{\alpha} e_3$

An action, α provides the labeling on an arrow, making it explicit what brings about a state-to-state transition.

As a shorthand for

(48) a. $(e_1, \alpha, e_2) \in \rightarrow$, we will also use:

b. $e_1 \xrightarrow{\alpha} e_3$



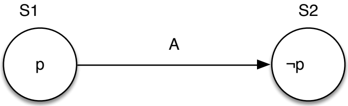
Dynamic Event Model

Labeled Transition System (LTS)

If reference to the state content (rather than state name) is required for interpretation purposes, then as shorthand for: $(\{\phi\}_{e_1}, \alpha, \{\neg\phi\}_{e_2}) \in \rightarrow$, we use:

If reference to the state content (rather than state name) is required for interpretation purposes, then as shorthand for: $(\{\phi\}_{e_1}, \alpha, \{\neg\phi\}_{e_2}) \in \rightarrow$, we use:

(50) $\boxed{\phi}_{e_1} \xrightarrow{\alpha} \boxed{\neg\phi}_{e_2}$

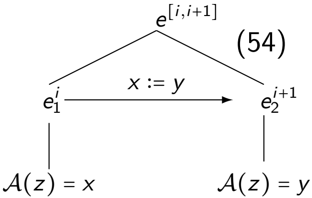


Simple First-order Transition

(51) $x := y$ (ν-transition)
"x assumes the value given to y in the next state."
$\langle \mathcal{M}, (i, i+1), (u, u[x/u(y)]) \rangle \models x := y$
iff $\langle \mathcal{M}, i, u \rangle \models s_1 \wedge \langle \mathcal{M}, i+1, u[x/u(y)] \rangle \models x = y$

Simple First-order Transition

(53) $x := y$ (ν-transition)
"x assumes the value given to y in the next state."
$\langle \mathcal{M}, (i, i+1), (u, u[x/u(y)]) \rangle \vDash x := y$
iff $\langle \mathcal{M}, i, u \rangle \vDash s_1 \wedge \langle \mathcal{M}, i+1, u[x/u(y)] \rangle \vDash x = y$



(54)

Processes

With a ν-transition defined, a *process* can be viewed as simply an iteration of basic variable assignments and re-assignments:

Processes

With a ν-transition defined, a *process* can be viewed as simply an iteration of basic variable assignments and re-assignments: