Model-Checking Event Structures

Madhavan Mukund

Chennai Mathematical Institute
http://www.cmi.ac.in/~madhavan

Formal Methods Update Meeting
IIT Roorkee
13 July 2009
Logics and verification

Temporal logic

- Used to describe properties to be verified
- Comes in two basic flavours
Temporal logic

- Used to describe properties to be verified
- Comes in two basic flavours

Linear-time temporal logic

- Interpreted separately over each possible run of the system
- To satisfy a property, every run must satisfy it
Logics and verification

Temporal logic

- Used to describe properties to be verified
- Comes in two basic flavours

Linear-time temporal logic

- Interpret separately over each possible run of the system
- To satisfy a property, every run must satisfy it

Branching-time temporal logic

- Collect all runs of the system in a single structure, the computation tree
- Interpret formulas over computation tree
- Can quantify over runs, compare runs, ...
The Computation Tree

- We will work finite-state systems
- Start at an initial state and explore all executions
- Unfold the system into a tree
The Computation Tree

- We will work finite-state systems
- Start at an initial state and explore all executions
- Unfold the system into a tree
The Computation Tree

- We will work finite-state systems
- Start at an initial state and explore all executions
- Unfold the system into a tree
Concurrent systems

- Suppose the system is a collection of interacting components
- During a run, some actions can be independent of each other
- Different sequences of actions may represent the same run
Concurrent systems

- Suppose the system is a collection of interacting components
- During a run, some actions can be independent of each other
- Different sequences of actions may represent the same run
Concurrent systems

- Suppose the system is a collection of interacting components
- During a run, some actions can be independent of each other
- Different sequences of actions may represent the same run

![Diagram of a concurrent system with labeled transitions: e1 e2 e3 e4 e5, e1 e2 e4 e3 e5, e1 e4 e2 e3 e5, e4 e1 e2 e3 e5]
Runs as partial orders

- Convenient to view each execution as a labelled partial order
- Actions can be related in two ways
  - Causality
    An occurrence of $b$ causally depends on an occurrence of $a$ if $a$ must happen before $b$ happens
  - Concurrency
    An occurrence of $b$ is independent of an occurrence of $a$ if they can occur in any order

\[ e_1 \rightarrow e_2 \rightarrow e_3 \]
\[ e_4 \rightarrow e_5 \]
Runs as partial orders

- In many interesting cases, the runs of concurrent systems can be described as traces [Mazurkiewicz]
- Actions are enriched with independence relation specifying which pairs are independent
  - Symmetric, irreflexive
  - Typically derived from structure of underlying system
    - Actions performed by disjoint sets of components
- In a linearization, adjacent independent actions can be swapped to yield an equivalent linearization

\[
\begin{align*}
  e_1 & \ e_2 & \ e_3 & \ e_4 & \ e_5 \\
  e_1 & \ e_2 & \ e_4 & \ e_3 & \ e_5 \\
  e_1 & \ e_4 & \ e_2 & \ e_3 & \ e_5 \\
  e_4 & \ e_1 & \ e_2 & \ e_3 & \ e_5
\end{align*}
\]
Temporal logics for concurrent systems

- Temporal logic interpreted on linearizations of runs makes too many distinctions
  - A property such as $e_2$ is immediately followed by $e_4$ is true in some linearizations and not in others
- Modify temporal logic to express causality and concurrency
- What about linear-time vs branching-time?
- How do we represent the computation tree of a concurrent system?
Computation tree of a concurrent system

- In sequential systems, computation tree glues together all runs in a single branching structure.
- In concurrent systems, each run is a labelled partial order (a trace).
- Need to glue together traces to form a tree.
Event structures

- Each action occurs in a context — what has happened earlier
- Each different occurrence of an action is an event
- In a single trace, events are related by causality or concurrency
- Across traces, events are related by conflict
  - Choosing between two mutually incompatible events generates different runs
Formally, an event structure is of the form $ES = (E, \leq, \#)$

- $E$ is the set of events
- $\leq$ is the causality relation (a partial order)
- $\#$ is a binary conflict relation
  - Irreflexive, symmetric
- Conflict is inherited via causality
  - $e \# f$ and $f \leq f'$ implies $e \# f'$
- Two events are concurrent if they are not related by $\leq$ or $\#$
  — $e \text{ co } f$
Event Structures . . .
Event Structures . . .

- A configuration of an event structure is a set $X \subseteq E$ such that
  - $X$ is \downarrow-closed
  - $X$ is conflict free

- A configuration represents a set of events that have happened so far — a global state of the system
From traces to event structures

- Can extract an event structure from the set of traces
From traces to event structures

- Can extract an event structure from the set of traces
- $t \leq t'$ if $t'$ extends $t$ with more events
  - For instance, $[e_1 e_2 e_3] \leq [e_1 e_4 e_2 e_3]$
From traces to event structures

- Can extract an event structure from the set of traces
- \( t \leq t' \) if \( t' \) extends \( t \) with more events
  - For instance, \([e_1 e_2 e_3] \leq [e_1 e_4 e_2 e_3]\)
- \( t \) and \( t' \) are compatible if there is \( t'' \) such that \( t \leq t'' \) and \( t' \leq t'' \)
  - For instance, \([e_1 e_2 e_3]\) and \([e_4]\) are compatible because both are dominated by \([e_1 e_2 e_3 e_4]\)
From traces to event structures

- Can extract an event structure from the set of traces
  - $t \leq t'$ if $t'$ extends $t$ with more events
    - For instance, $[e_1e_2e_3] \leq [e_1e_4e_2e_3]$
  - $t$ and $t'$ are compatible if there is $t''$ such that $t \leq t''$ and $t' \leq t''$
    - For instance, $[e_1e_2e_3]$ and $[e_4]$ are compatible because both are dominated by $[e_1e_2e_3e_4]$
  - $t \not\approx t'$ if $t$ and $t'$ are not compatible
Can extract an event structure from the set of traces

- \( t \leq t' \) if \( t' \) extends \( t \) with more events
  - For instance, \([e_1e_2e_3] \leq [e_1e_4e_2e_3]\)

- \( t \) and \( t' \) are compatible if there is \( t'' \) such that \( t \leq t'' \) and \( t' \leq t'' \)
  - For instance, \([e_1e_2e_3] \) and \([e_4] \) are compatible because both are dominated by \([e_1e_2e_3e_4]\)

- \( t \# t' \) if \( t \) and \( t' \) are not compatible

- Identify events with prime traces
  - Prime trace: Only one maximal element
  - “Earliest” occurrence of an action
Temporal logics for event structures

- Interpret formulas at events of an event structure
  - Event $e$ denotes the minimal configuration $\downarrow e$ where it occurs
  - $\downarrow e$ can be thought of as the local state of the components involved in $e$

- Modalities to express causality, conflict, concurrency
  - $ES, e \models A_{\leq} \varphi$ if at every $f$ such that $e \leq f$, $ES, f \models \varphi$
  - $ES, e \models E_{\leq} \varphi$ if there exists $f$ such that $e \leq f$ and $ES, f \models \varphi$
  - $ES, e \models A_{\#} \varphi$ if at every $f$ such that $e \# f$, $ES, f \models \varphi$
  - $ES, e \models E_{\#} \varphi$ if there exists $f$ such that $e \# f$ and $ES, f \models \varphi$
  - $ES, e \models A_{co} \varphi$ if at every $f$ such that $e \ co f$, $ES, f \models \varphi$
  - $ES, e \models E_{co} \varphi$ if there exists $f$ such that $e \ co f$ and $ES, f \models \varphi$
The immediate successor relation $\prec$ generates $\preceq$

$$ e \prec f \text{ if } e \leq f \text{ and for all } g, \text{ } e \leq g \leq f \Rightarrow e = g \text{ or } g = f $$

Corresponding modality

- $ES, e \models A_{\prec} \varphi$ if at every $f$ such that $e \prec f$, $ES, f \models \varphi$
- $ES, e \models E_{\prec} \varphi$ if there exists $f$ such that $e \prec f$ and $ES, f \models \varphi$

Minimal conflict relation $\#_{\mu}$ generates $\#$ via $\preceq$

$$ e \#_{\mu} f \text{ if } e \# f \text{ and for all } e' \prec e, f' \prec f, \text{ it is not the case that } e \# f' \text{ or } e' \# f $$

Corresponding modality

- $ES, e \models A_{\#_{\mu}} \varphi$ if at every $f$ such that $e \#_{\mu} f$, $ES, f \models \varphi$
- $ES, e \models E_{\#_{\mu}} \varphi$ if there exists $f$ such that $e \#_{\mu} f$ and $ES, f \models \varphi$
The model-checking problem

- Start with a finite-state representation of the system, $M$
  - Petri net
  - Product of automata
  - …
- Atomic propositions $AP$ describing properties of (local) states
  - Valuation assigns a subset of $AP$ to each (local) state
- In $ES_M$, event structure of $M$, each configuration $↓e$ corresponds to a unique (global) state of $M$
- Formulas for system properties built from $AP$, boolean operations, event structure modalities
  - $A ≤ (b \Rightarrow E_# a)$
  - $A < ((b \Rightarrow E_# µ a) \land (a \Rightarrow E_# µ b))$
- Does $ES_M, e \models \varphi$?
The model-checking problem

Apparently only two papers addressing this topic.

- Model-Checking for a Subclass of Event Structures
  W Penczek
  TACAS 1997

- Model-Checking Trace Event Structures
  P Madhusudan
  LICS 2003
The model-checking problem

Apparently only two papers addressing this topic.

- Model-Checking for a Subclass of Event Structures
  W Penczek
  TACAS 1997
  Today

- Model-Checking Trace Event Structures
  P Madhusudan
  LICS 2003
  Tomorrow
Running example
Trace semantics
Event structure semantics
Penczek’s result

- Restrict the modalities
- Only $A_{\leq}/E_{\leq}$, $A_{<}/E_{<}$, $A_{#m\mu}/E_{#\mu}$
- No $A_{\text{co}}/E_{\text{co}}$
- Though $#$ is generated by $#_{\mu}$ and $\leq$, cannot fully express $A_{#}/E_{#}$
  - $#$ = $\leq^{-1} \circ #_{\mu} \circ \leq$
  - No past modality to capture $\leq^{-1}$
Quotienting the trace system

First attempt

▷ Two traces are equivalent if they reach the same global state
▷ This is not sufficient,
  ▷ [], [agbc] and [hdef] all lead to (1, 2, 7)
  ▷ Causal futures are different

Refine further

▷ Two traces are equivalent if they reach the same global state and the maximal actions in the traces are the same
Quotienting the trace system . . .

- Recall that events are prime traces
- Let $e$, $f$ be two events
- Would like the following property
  - If $[e] = [f]$, then $ES_M, e \models \varphi$ iff $ES_M, f \models \varphi$
- This does not hold in general
Another example
Quotienting the trace system . . .

- Recall that events are prime traces
- Let $e$, $f$ be two events
- Would like the following property
  - If $[e] = [f]$, then $ES_M, e \models \varphi$ iff $ES_M, f \models \varphi$
- This does not hold in general
  - In this example, $[bd]$ and $[cd]$ are prime traces reaching same global state $(1, 7, 8)$ with same maximal event $d$
  - $[a]$ and $[ba]$ are prime traces that reach different global states
  - $[bd] \#_\mu[ba]$ and $[cd] \#_\mu[a]$, so $[bd]$ and $[cd]$ don’t satisfy the same $E_{\#_\mu}$ formulas.
Free-choice property

- Restrict systems to have the free-choice property
- Intuitively, choices available to one component cannot change due to actions of another component
  - Second example is not free-choice — if third process executes \( c \), second process loses option of performing \( b \) (and vice versa)

Formally

- If \( s \xrightarrow{a} s' \) involves components \( P_a \) and \( t \xrightarrow{b} t' \) involves components \( P_b \) then
  - Either \( P_a \cap P_b = \emptyset \), or,
  - If \( s[j] = t[j] \) for some process \( j \), then \( P_a = P_b \) and for all \( j \in P_a = P_b \), \( s[j] = t[j] \).
Free-choice property . . .

- Free-choice ensure that if $e \#_{\mu} f'$ then
  - There is a common prefix $t$ such that $e = ta$, $f = tb$
  - Actions $a$ and $b$ involve exactly the same set of components

Recall our definition of a quotiented trace system

$t \equiv t'$ if $t$ and $t'$ reach the same global state and have the same set of maximal actions

Our desired property

- If $e \equiv f$, then $ES_M, e \models \varphi$ iff $ES_M, f \models \varphi$

can be proved by structural induction on $\varphi$ from the following

- If $e \equiv f$ and $e < e'$, there exists $f'$ such that $f < f'$ and $e' \equiv f'$
- If $e \equiv f$ and $e \#_{\mu} e'$, there exists $f'$ such that $f \#_{\mu} f'$ and $e' \equiv f'$
The model-checking algorithm

Naive

- Compute the quotiented trace system for the given model
  - This is a finite object
- Identify prime traces as events and constructed corresponding “quotiented” event structure
- Decompose formulas and use CTL-style bottom-up labelling to identify the formulas true at each event

More efficient

- Use partial order techniques to construct a representative subset of the quotiented trace system
- Translate event structure logic to CTL and directly use CTL model checking on the quotiented trace system
  - Use a special proposition to mark prime traces
Beyond Penczek’s result

- Can we extend the algorithm to full logic involving $A_{co}/E_{co}$ and $A_{#}/E_{#}$?
- Can the free-choice restriction be relaxed?