CS 6156

LTL Monitor Synthesis

Owolabi Legunsen

Fall 2020

Logistics

- Project proposals are due 10/6 AoE
- Homework 1 (likely) released next week

From last lecture…

- ptLTL monitor synthesis
- Monitoring with Maude and a monitor synthesis algorithm
- Can you write your own generic monitoring algorithm for ptLTL without synthesizing monitors?

In this lecture…

- LTL syntax and semantics
- Intro to BDDs
- “Special” FSMs that LTL specs get translated to
- Algorithms for translating LTL to FSMs

In this lecture…

- LTL syntax and semantics
- Intro to BDDs
- “Special” FSMs that LTL specs get translated to
- Algorithms for translating LTL to FSMs

Not in this lecture

- Asynchronous Maude “interpreter” monitoring algorithm
- Synthesis of dynamic-algorithm monitors for LTL
  - It’s like the one in the last class on ptLTL
  - But it traverses traces backwards, i.e., it works asynchronously
- A more efficient “online” monitoring algorithm in Maude

LTL Syntax

$$\varphi \coloneqq \mathrm{p} \mid (\varphi) \mid \neg\varphi \mid \varphi \wedge \varphi' \mid \varphi \vee \varphi' \mid \circ\varphi \mid \varphi\, \mathcal{U}\, \varphi' \mid \Box\varphi \mid \Diamond\varphi$$

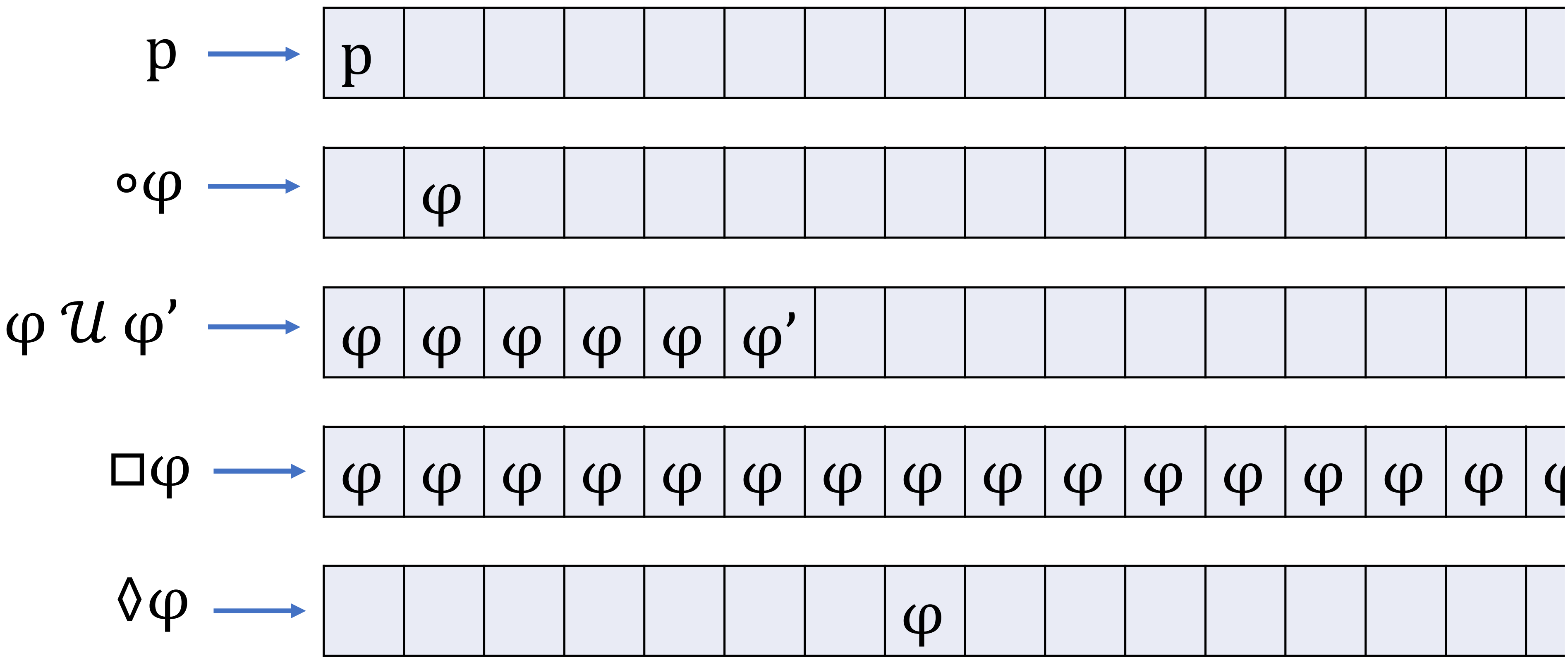
- p – a proposition over state (event) variables
- $\circ\varphi$ – “next”
- $\varphi\, \mathcal{U}\, \varphi'$ – “until”
- $\Box\varphi$ – “always”, “forever”, “box”
- $\Diamond\varphi$ – “eventually”, “sometime”, “diamond”

LTL standard model

$t : \mathbb{N}^{+} \rightarrow 2^{\mathcal{P}}$ for some set of atomic propositions $\mathcal{P}$

- t maps each time point to the set of propositions that hold at that point

LTL Semantics (informally)



Finite trace future time LTL semantics

- In RV, we only have finite traces. So we need a different semantics over finite traces

- Finite trace t: a non-empty finite sequence of states, each state denoting the set of propositions that hold at that state
  - State == Event?

Finite trace future time LTL prelims

- head (e, t) = head (e) = e
- tail (e, t) = t
- length(e) = 1
- length(e, t) = 1 + length(t)
- t_i : suffix of trace t that starts at position i

τ

Finite trace future time LTL (1)

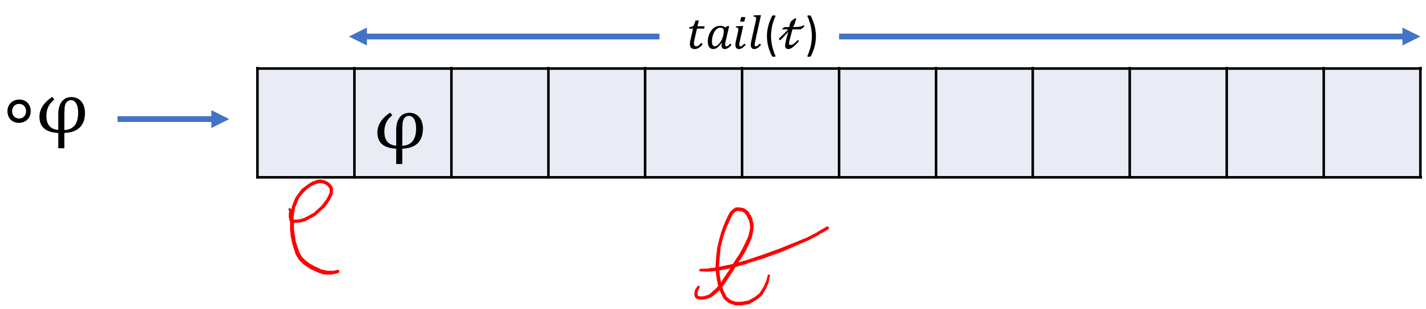
$t \vDash$ f when a trace t satisfies a formula f, defined as

$t \vDash \mathrm{true}$	iff	*true*
$t \vDash \mathrm{false}$	iff	*false*
$t \vDash \mathrm{p}$	iff	$\mathrm{p} \in head(t)$
$t \vDash \varphi \wedge \varphi'$	iff	$t \vDash \varphi$ and $t \vDash \varphi'$
$t \vDash \varphi \mathbin{++} \varphi'$	iff	$t \vDash \varphi$ xor $t \vDash \varphi'$
$t \vDash \circ\varphi$	iff	**if** $tail(t)$ is defined **then** $tail(t) \vDash \varphi$ **else** $t \vDash \varphi$
$t \vDash \Diamond\varphi$	iff	$(\exists i \leq length(t))\ t_{\mathrm{i}} \vDash \varphi$
$t \vDash \Box\varphi$	iff	$(\forall i \leq length(t))\ t_{\mathrm{i}} \vDash \varphi$
$t \vDash \varphi\ \mathcal{U}\ \varphi'$	iff	$(\exists i \leq length(t))\ (t_{\mathrm{i}} \vDash \varphi'$ and $(\forall \mathrm{j} < i)\ t_{\mathrm{j}} \vDash \varphi)$

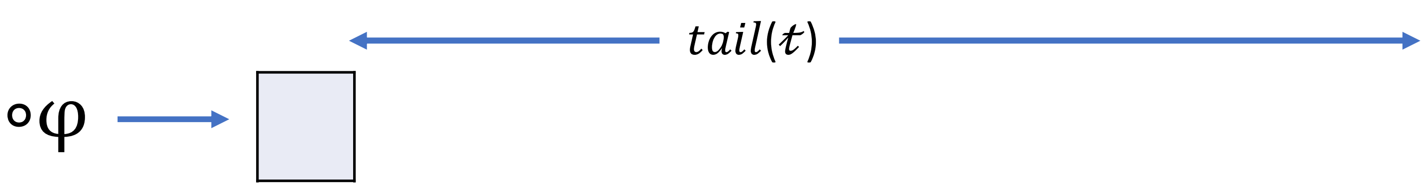
Finite trace future time LTL (2)

$t \vDash \circ\varphi$	iff	**if** $tail(t)$ is defined **then** $tail(t) \vDash \varphi$ **else** $t \vDash \varphi$

Case 1: $tail(t)$ is defined

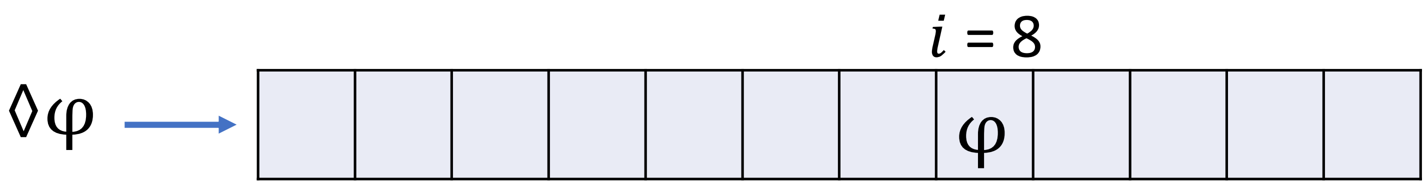


Case 2: $tail(t)$ is not defined



Finite trace future time LTL (3)

$t \vDash \lozenge\varphi$	iff	$(\exists i \leq length(t))\ t_{\mathrm{i}} \vDash \varphi$



Recall: t_{i} is the suffix of trace t that starts at position i

Finite trace future time LTL (4)

$t \vDash \varphi\,\mathcal{U}\,\varphi'$	iff	$(\exists i \leq length(t))\ (t_i \vDash \varphi'$ and $(\forall j < i)\ t_j \vDash \varphi)$

$i = 6$

$\varphi\,\mathcal{U}\,\varphi' \longrightarrow$

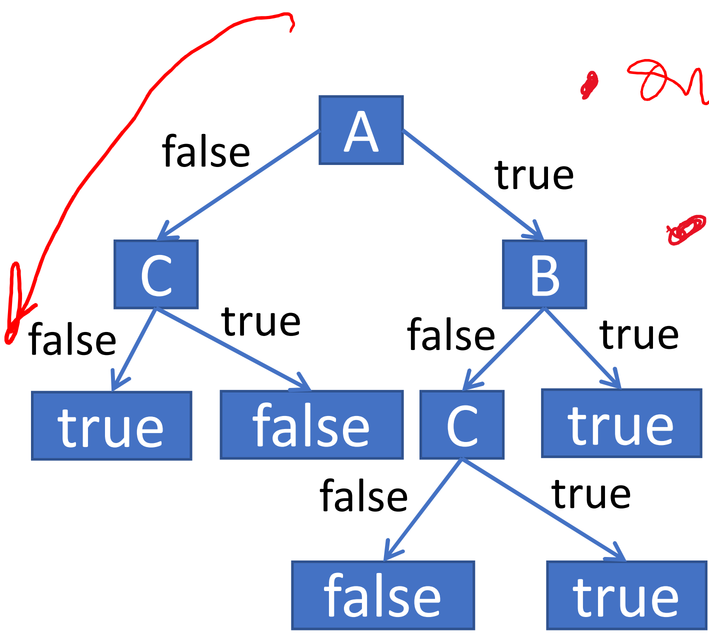
φ	φ	φ	φ	φ	φ’						

Recall: t_i is the suffix of trace t that starts at position i

Binary Decision Diagrams: examples

Given ((A ∧ B) ∨ ¬C), where A, B, C are propositions

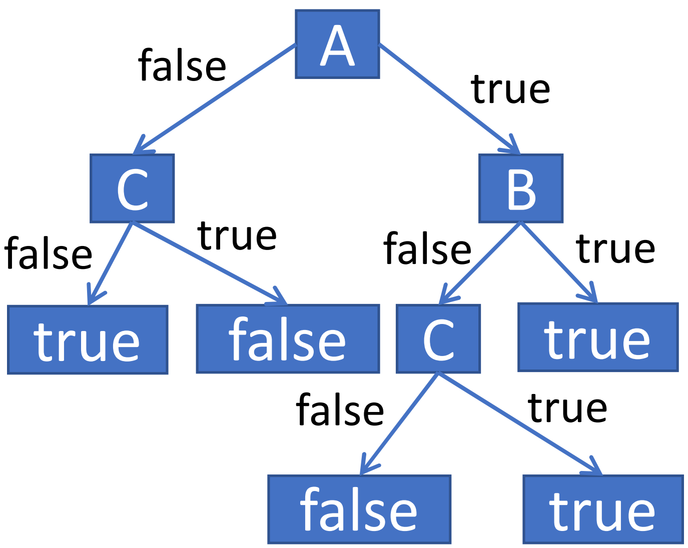
- each proposition may appear multiple
- only 2 possible values @ leaves "true" "false" nodes
- it's a tree
- child nodes are
- edges are labeled "true", "false"
- it doesn't have to be balanced
- paths are valuations

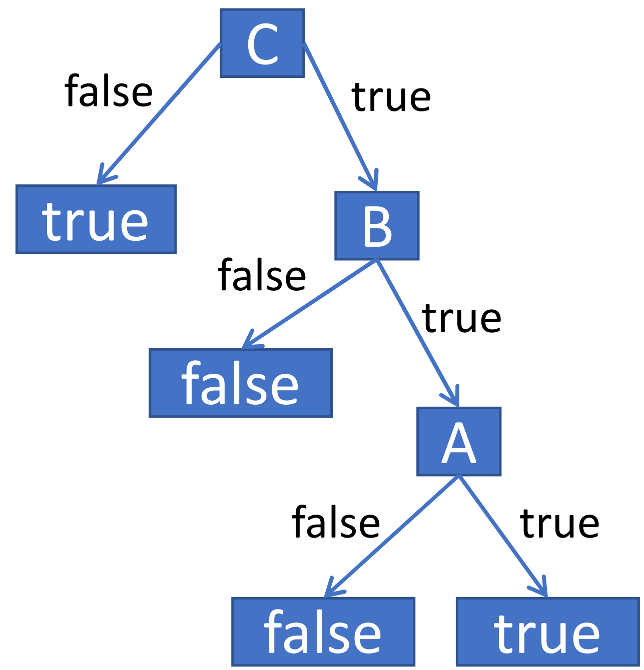


What do you notice about this BDD?

Binary Decision Diagrams: examples

Given $((A \wedge B) \vee \neg C)$, where A, B, C are propositions





What do you notice about this BDD?

Some things to know about BDDs

- A way to represent Boolean formulas

- A formula can have many BDD representations

- Problem: find a BDD that is "most efficient"
  - The order of propositions in the BDD is important

- Procedures exist for
  - creating a BDD from a formula
  - creating a Reduced-order BDD from a BDD

Our LTL monitor-synthesis goal

- Synthesize an FSM that receives an event θ and transitions as fast as possible to a new state
- We will explore two such FSMs
  - Multi-transition FSMs
  - Binary-transition tree FSMs
- What is a multi-transition?
- What is binary-transition tree?

Multi-transitions

- Let S be a set of states s.t. $\{s_1, s_2, \ldots, s_n\} \in S$
- Let A be a set of atomic predicates s.t.
  - $p_1, p_2, \ldots, p_n$ are propositions over atoms in A
  - $p_1 \vee p_2 \vee \ldots \vee p_n$ holds
  - for any distinct p_i and p_j, $p_i \rightarrow \neg p_j$
- Then, $[p_1 ? s_1, p_2 ? s_2, \ldots, p_n ? s_n]$ is a **multi-transition (MT)** over S and A
- MT(S, A) is the set of MTs over S and A

Multi transitions on events

- Let θ be an event
- Then θ_{MT} is a function that maps MTs to states after θ is received

$$\theta_{\mathrm{MT}}([p_1 ? s_1, p_2 ? s_2, \ldots, p_n ? s_n]) = s_i \text{ if } \theta(p_i) = \mathit{true}$$

Binary Transition Trees (BTTs)

- Syntax
  - $\mathrm{BTT} \coloneqq$ S | (A ? BTT: BTT)
- Let BTT(S, A) be the set of BTTs over S and A
- Then θ_{BTT} is a function that maps BTTs to states after event θ is received

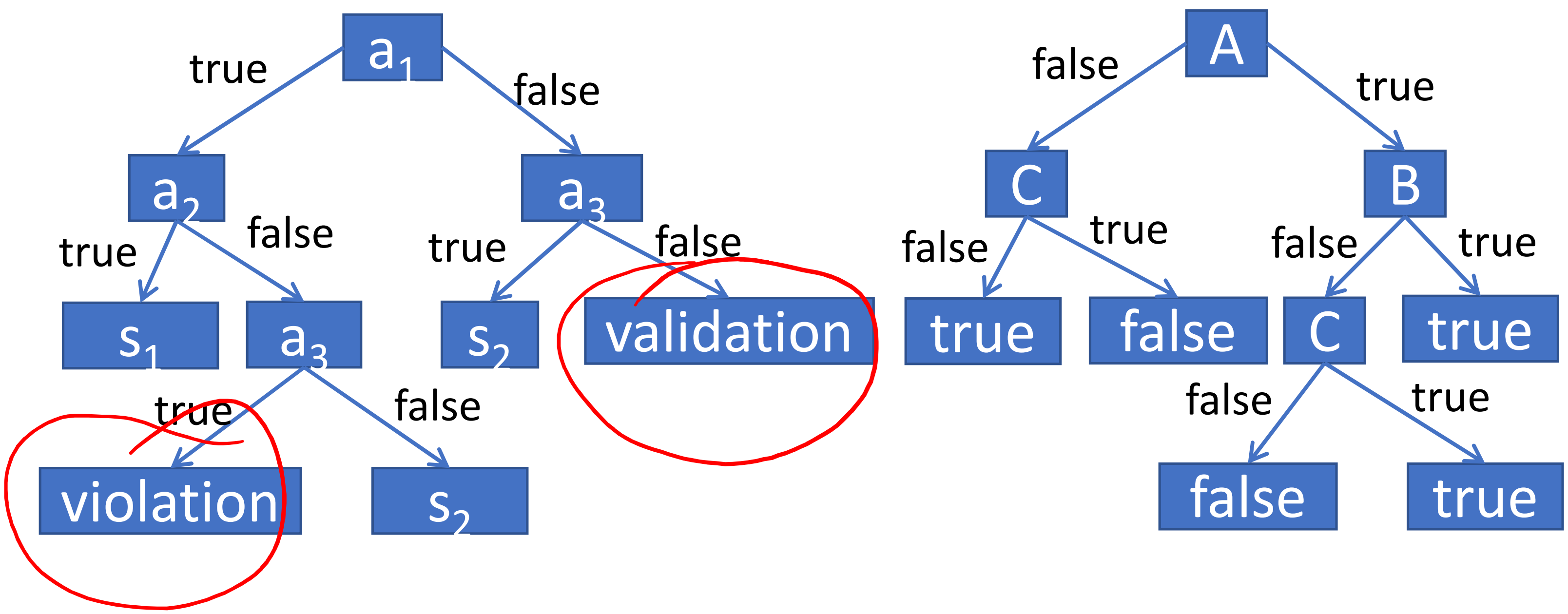
$$\theta_{BTT}(s) = s \text{ for any } s \in S,$$
$$\theta_{BTT}(a \mathbin{?} b_1 : b_2) = \theta_{BTT}(b_1) \text{ if } \theta(a) \text{ is } \textit{true}, \text{ and}$$
$$\theta_{BTT}(a \mathbin{?} b_1 : b_2) = \theta_{BTT}(b_2) \text{ if } \theta(a) \text{ is } \textit{false}$$

Relating MTs and BTTs

A BTT b in $BTT(S, A)$ *implements* a MT t in $MT(S, A)$ iff $\theta_{BTT}(b) = \theta_{MT}(t)$ for any event θ

BTT Example

$a_1 ? a_2 ? s_1 : a_3 ? \text{violation} : s_2 : a_3 ? s_2 : \text{validation}$



BTTs as generalizations of BDDs?

Recall: our goal

- Synthesize an FSM that receives an event θ and transitions as fast as possible to a new state

- We will explore two such FSMs
  - Multi-transition FSMs
  - Binary-transition tree FSMs

- ~~What is a multi-transition?~~

- ~~What is binary-transition tree?~~

MT-FSM

- An MT-FSM is a triple (S, A, μ), where S is a set of states, A is a set of atomic predicates, and μ is a map from S – {violation, validation} to MT(S, A).
    - In a terminating MT-FSM*, μ* maps to MT({violation, validation} , A).
- If we reach {violation, validation}, stay there
- On event θ, transition $s \xrightarrow{\theta} s'$ denotes $\theta_{\mathrm{MT}}(\mu(s)) = s'$

MT-FSM by example

$p_1 ? s_1, p_2 ? s_2 \sim$

State	MT for non-terminal events	MT for terminal events
1	[yellow \/ !green ? 1, !yellow /\ green /\ !red ? 2, !yellow /\ green /\ red ? *false*]	[yellow \/ !green ? *true*, !yellow /\ green ? *false*]
2	[yellow ? 1, !yellow /\ !red ? 2, !yellow /\ red ? *false*]	[yellow ? *true*, !yellow ? *false*]

Figure 3: MT-FSM for the formula `[](green -> !red U yellow)`.

BTT-FSM

- A BTT-FSM is a triple (S, A, β), where S is a set of states, A is a set of atomic predicates, and β is a map from S – {violation, validation} to BTT(S, A).
  - In a terminating BTT-FSM*, β* maps to BTT({violation, validation} , A).
- If we reach {violation, validation}, stay there
- On event θ, transition $s \xrightarrow{\theta} s'$ denotes $\theta_{\mathrm{BTT}}(\beta(s)) = s'$

BTT-FSM by example

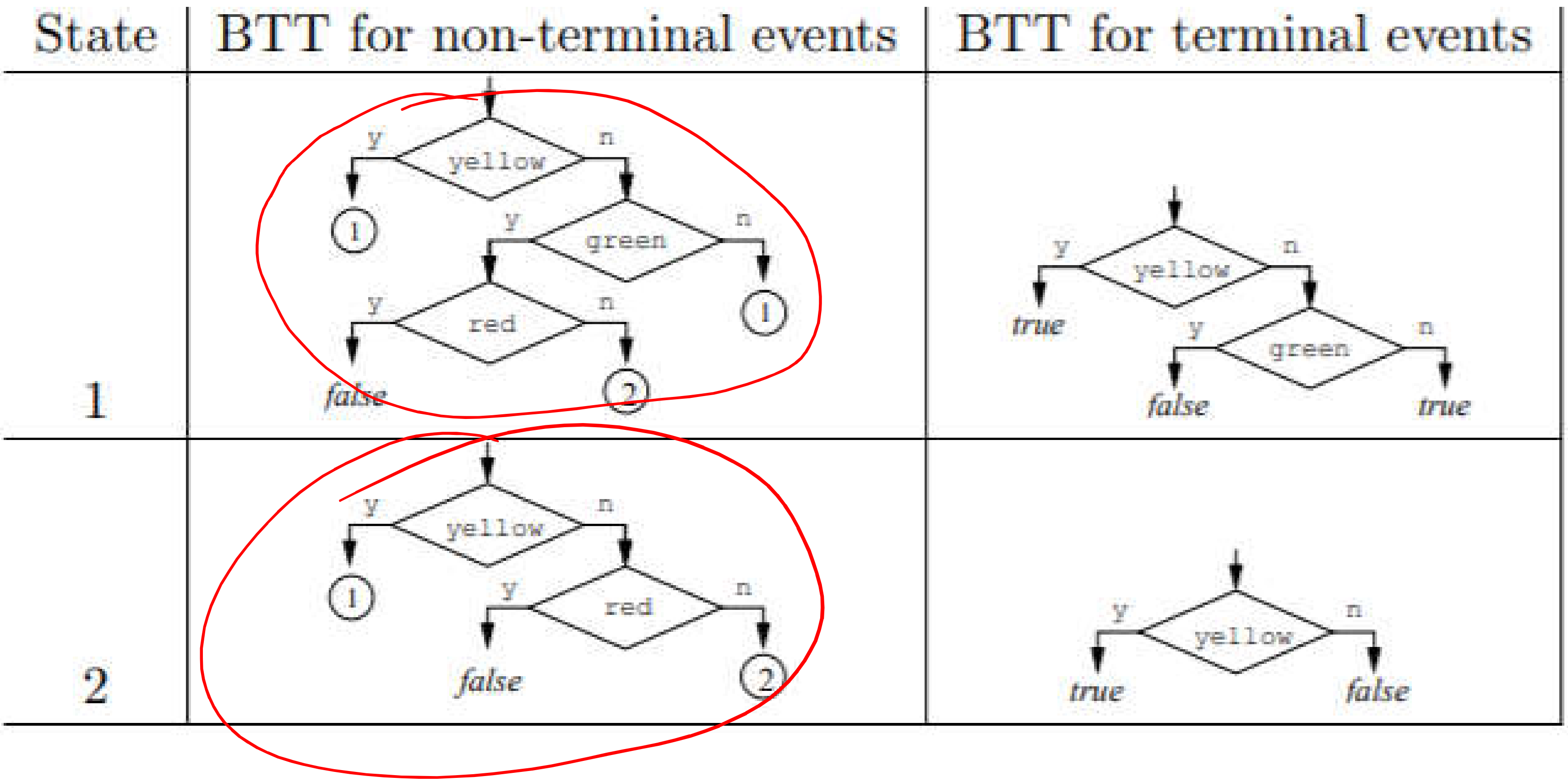


Figure 4: A BTT-FSM for the formula `[](green -> !red U yellow)`.

BTT-FSMs are efficient MT-FSMs

- One way to think about it informally
  - BTT-FSMs are to MT-FSMs what RoBDDs are to BDDs

- Another way to think about it informally
  - MT-FSMs: many if-then statements, all conditions evaluated
  - BTT-FSMs: if-then-else sequence, only some conditions usually need to be evaluated

- LTL synthesis: LTL spec → MT-FSM → BTT-FSM

Why not LTL → BTT-FSMs?

- Short answer: state mergeability is well defined and allows for more elegant LTL → MT-FSM conversion

$\textsc{Merge}([p_1?s_1, p_2?s_2, ..., p_n?s_n], [p'_1?s'_1, p'_2?s'_2, ..., p'_n?s'_{n'}])$
contains all choices $p?s''$, where s'' is a state in $\{s_1, s_2, ..., s_n\} \cup \{s'_1, s'_2, ..., s'_{n'}\}$ and

- p is p_i when $s'' = s_i$ for some $1 \leq i \leq n$ and $s'' \neq s'_{i'}$ for all $1 \leq i' \leq n'$, or
- p is $p'_{i'}$ when $s'' = s'_{i'}$ for some $1 \leq i' \leq n'$ and $s'' \neq s_i$ for all $1 \leq i \leq n$, or
- p is $p_i \vee p'_{i'}$ when $s'' = s_i$ for some $1 \leq i \leq n$ and $s'' = s'_{i'}$ for some $1 \leq i' \leq n'$.

- MERGE is used in the LTL2MT-FSM algorithm
- Is this elegance at the cost of efficiency?

MT-FSM → BTT-FSM conversion

- Recall: many BDDs can represent the same formula
- Similarly, many BTTs can represent the same MT
- How do we find optimal BTT for an MT?
  - Enumerate BTT all and pick the most efficient
- Is it time to revisit the optimal BTT problem (probabilities, cost, new algorithm)?

Zoom break (3 minutes)

LTL spec → MT-FSM preliminaries

φ → LTL2MT-FSM → MT-FSM

□(green → ¬red $\mathcal{U}$ yellow)

State	MT for non-terminal events	MT for terminal events
1	[yellow \/ !green ? 1, !yellow /\ green /\ !red ? 2, !yellow /\ green /\ red ? *false*]	[yellow \/ !green ? *true*, !yellow /\ green ? *false*]
2	[yellow ? 1, !yellow /\ !red ? 2, !yellow /\ red ? *false*]	[yellow ? *true*, !yellow ? *false*]

MT-FSM states are formulas

φ contains all event names

Key idea: After event θ occurs, what formulas can φ be re-written to?

Terminal states: what if θ is the last event in the trace?

Formula rewriting basics

- Intuition:
  - Let trace t = E, T consist of event E followed by trace T
  - Formula X holds on t iff X{E} holds on T
  - If E is terminal, then X{E*} holds iff X holds in standard LTL semantics
- Rewrite rules:

```
eq (o X){E} = X .
eq (o X){E *} = X{E *} .
eq (<> X){E} = X{E} \/ <> X .
eq (<> X){E *} = X{E *} .
eq ([] X){E}   = X{E} /\ [] X .
eq ([] X){E *} = X{E *} .
eq (X U Y){E} = Y{E} \/ X{E} /\ X U Y .
eq (X U Y){E *} = Y{E *} .
op _|-_ : Trace Formula -> Bool .
eq E   |- X = [X{E *}] .
eq E,T |- X = T |- X{E} .
```

X must hold now (X{E}) and in the future (□X)

Formula rewriting example (1)

- Let X = □(green → ¬red $\mathcal{U}$ yellow), E = green yellow
- X =*=> □(true ++ green ++ green ∧ (true ++ red) $\mathcal{U}$ yellow)

```
([](true ++ green ++ green /\ (true ++ red) U yellow)){green yellow}   =*=>
(true ++ green{green yellow}
   ++ green{green yellow} /\ ((true ++ red) U yellow){green yellow})
     /\ [](true ++ green ++ green /\ (true ++ red) U yellow)           =*=>
((true ++ red) U yellow){green yellow})
     /\ [](true ++ green ++ green /\ (true ++ red) U yellow)           =*=>
(yellow{green yellow} \/ ((true ++ red{green yellow}) /\ (true ++ red) U yellow)
     /\ [](true ++ green ++ green /\ (true ++ red) U yellow)           =*=>
[](true ++ green ++ green /\ (true ++ red) U yellow)
```

Formula rewriting example (2)

- Let X = □(green → ¬red $\mathcal{U}$ yellow), E = green
- X =*=> □(true ++ green ++ green ∧ (true ++ red) $\mathcal{U}$ yellow)

```
([](true ++ green ++ green /\ (true ++ red) U yellow)){green}        =*=>
(true ++ green{green}
   ++ green{green} /\ ((true ++ red) U yellow){green})
     /\ [](true ++ green ++ green /\ (true ++ red) U yellow)          =*=>
((true ++ red) U yellow){green})
     /\ [](true ++ green ++ green /\ (true ++ red) U yellow)          =*=>
(yellow{green} \/ ((true ++ red{green}) /\ (true ++ red) U yellow)
     /\ [](true ++ green ++ green /\ (true ++ red) U yellow)          =*=>
(true ++ red) U yellow /\ [](true ++ green ++ green /\ (true ++ red) U yellow)
```

Rewriting takes many steps! Sections 4.2 and 6.1 have details

Theorem 2: rewriting terminates (among other things)

LTL2MT-FSM algorithm (1)

let S **be** φ
procedure LTL2MT-FSM(φ)
  let $\mu^{\star}(\varphi)$ **be** $\emptyset$
  let $\mu(\varphi)$ **be** $\emptyset$
  foreach $\theta : A \rightarrow \{\mathit{true}, \mathit{false}\}$ **do**
    let e_θ **be** the list of atoms a with $\theta(a) = \mathit{true}$
    let p_θ **be** the proposition $\bigwedge\{a \mid \theta(a) = \mathit{true}\} \wedge \bigwedge\{\neg a \mid \theta(a) = \mathit{false}\}$
    let $\mu^{\star}(\varphi)$ **be** $\textsc{Merge}([p_\theta \ ? \ \varphi\{e^{\star}_{\theta}\}], \mu^{\star}(\varphi))$
    let φ_θ **be** $\varphi\{e_\theta\}$
    if there is $\varphi' \in S$ with $\textsc{Valid}(\varphi_\theta \leftrightarrow \varphi')$
    then let $\mu(\varphi)$ **be** $\textsc{Merge}([p_\theta \ ? \ \varphi'], \mu(\varphi))$
    else let S **be** $S \cup \{\varphi_\theta\}$
      let $\mu(\varphi)$ **be** $\textsc{Merge}([p_\theta \ ? \ \varphi_\theta], \mu(\varphi))$
      LTL2MT-FSM(φ_θ)
  endfor
  if $\mu(\varphi) = [\mathit{true} \ ? \ \varphi]$ and $\mu^{\star}(\varphi) = [\mathit{true} \ ? \ b]$ **then** replace φ by b everywhere
endprocedure

Figure 5: Algorithm to generate a minimal MT-FSM* $(S, A, \mu, \mu^{\star}, \varphi)$ from an LTL formula φ.

LTL2MT-FSM algorithm (2)

let S **be** φ
procedure LTL2MT-FSM(φ)
  let $\mu^\star(\varphi)$ **be** $\emptyset$
  let $\mu(\varphi)$ **be** $\emptyset$
  foreach $\theta : A \rightarrow \{true, false\}$ **do**
    let e_θ **be** the list of atoms a with $\theta(a) = true$
    let p_θ **be** the proposition $\bigwedge\{a \mid \theta(a) = true\} \wedge \bigwedge\{\neg a \mid \theta(a) = false\}$
    let $\mu^\star(\varphi)$ **be** $\textsc{Merge}([p_\theta \ ?\ \varphi\{e_\theta^\star\}], \mu^\star(\varphi))$
    let φ_θ **be** $\varphi\{e_\theta\}$
    if there is $\varphi' \in S$ with $\textsc{Valid}(\varphi_\theta \leftrightarrow \varphi')$
    then let $\mu(\varphi)$ **be** $\textsc{Merge}([p_\theta \ ?\ \varphi'], \mu(\varphi))$
    else let S **be** $S \cup \{\varphi_\theta\}$
      let $\mu(\varphi)$ **be** $\textsc{Merge}([p_\theta \ ?\ \varphi_\theta], \mu(\varphi))$
      LTL2MT-FSM(φ_θ)
  endfor
  if $\mu(\varphi) = [true\,?\,\varphi]$ and $\mu^\star(\varphi) = [true\,?\,b]$ **then** replace φ by b everywhere
endprocedure

S is the set of states (initialized to {φ})

LTL2MT-FSM algorithm (3)

let S **be** φ
procedure LTL2MT-FSM(φ)
  let $\mu^\star(\varphi)$ **be** $\emptyset$
  let $\mu(\varphi)$ **be** $\emptyset$
  foreach $\theta : A \to \{\mathit{true}, \mathit{false}\}$ **do**
    let e_θ **be** the list of atoms a with $\theta(a) = \mathit{true}$
    let p_θ **be** the proposition $\bigwedge\{a \mid \theta(a) = \mathit{true}\} \wedge \bigwedge\{\neg a \mid \theta(a) = \mathit{false}\}$
    let $\mu^\star(\varphi)$ **be** $\textsc{Merge}([p_\theta \ ? \ \varphi\{e^\star_\theta\}], \mu^\star(\varphi))$
    let φ_θ **be** $\varphi\{e_\theta\}$
    if there is $\varphi' \in S$ with $\textsc{Valid}(\varphi_\theta \leftrightarrow \varphi')$
    then let $\mu(\varphi)$ **be** $\textsc{Merge}([p_\theta \ ? \ \varphi'], \mu(\varphi))$
    else let S **be** $S \cup \{\varphi_\theta\}$
      let $\mu(\varphi)$ **be** $\textsc{Merge}([p_\theta \ ? \ \varphi_\theta], \mu(\varphi))$
      LTL2MT-FSM(φ_θ)
  endfor
  if $\mu(\varphi) = [\mathit{true}\,?\,\varphi]$ and $\mu^\star(\varphi) = [\mathit{true}\,?\,b]$ **then** replace φ by b everywhere
endprocedure

For each state formula φ in S, maintain terminal (μ*(φ)) and non-terminal (μ(φ)) states

LTL2MT-FSM algorithm (4)

let S **be** φ
procedure LTL2MT-FSM(φ)
  let $\mu^\star(\varphi)$ **be** $\emptyset$
  let $\mu(\varphi)$ **be** $\emptyset$
  foreach $\theta : A \to \{\mathit{true}, \mathit{false}\}$ **do**
    let e_θ **be** the list of atoms a with $\theta(a) = \mathit{true}$
    let p_θ **be** the proposition $\bigwedge\{a \mid \theta(a) = \mathit{true}\} \wedge \bigwedge\{\neg a \mid \theta(a) = \mathit{false}\}$
    $\boxed{\textbf{let } \mu^\star(\varphi) \textbf{ be } \mathrm{MERGE}([p_\theta \mathrel{?} \varphi\{e^\star_\theta\}], \mu^\star(\varphi))}$
    let φ_θ **be** $\varphi\{e_\theta\}$
    if there is $\varphi' \in S$ with VALID($\varphi_\theta \leftrightarrow \varphi'$)
    then let $\mu(\varphi)$ **be** MERGE($[p_\theta \mathrel{?} \varphi'], \mu(\varphi)$)
    else let S **be** $S \cup \{\varphi_\theta\}$
      let $\mu(\varphi)$ **be** MERGE($[p_\theta \mathrel{?} \varphi_\theta], \mu(\varphi)$)
      LTL2MT-FSM(φ_θ)
  endfor
  if $\mu(\varphi) = [\mathit{true} \mathrel{?} \varphi]$ and $\mu^\star(\varphi) = [\mathit{true} \mathrel{?} b]$ **then** replace φ by b everywhere
endprocedure

Update μ*(φ) by considering θ to be the last event

LTL2MT-FSM algorithm (5)

let S **be** φ
procedure LTL2MT-FSM(φ)
  let $\mu^\star(\varphi)$ **be** $\emptyset$
  let $\mu(\varphi)$ **be** $\emptyset$
  foreach $\theta : A \rightarrow \{true, false\}$ **do**
    let e_θ **be** the list of atoms a with $\theta(a) = true$
    let p_θ **be** the proposition $\bigwedge\{a \mid \theta(a) = true\} \wedge \bigwedge\{\neg a \mid \theta(a) = false\}$
    let $\mu^\star(\varphi)$ **be** $\textsc{Merge}([p_\theta \mathbin{?} \varphi\{e^\star_\theta\}], \mu^\star(\varphi))$
    let φ_θ **be** $\varphi\{e_\theta\}$
    if there is $\varphi' \in S$ with $\textsc{Valid}(\varphi_\theta \leftrightarrow \varphi')$
    then let $\mu(\varphi)$ **be** $\textsc{Merge}([p_\theta \mathbin{?} \varphi'], \mu(\varphi))$
    else let S **be** $S \cup \{\varphi_\theta\}$
      let $\mu(\varphi)$ **be** $\textsc{Merge}([p_\theta \mathbin{?} \varphi_\theta], \mu(\varphi))$
      LTL2MT-FSM(φ_θ)
  endfor
  if $\mu(\varphi) = [true \mathbin{?} \varphi]$ and $\mu^\star(\varphi) = [true \mathbin{?} b]$ **then** replace φ by b everywhere
endprocedure

Rewrite φ to φ{θ}

LTL2MT-FSM algorithm (6)

let S **be** φ
procedure LTL2MT-FSM(φ)
    let $\mu^\star(\varphi)$ **be** $\emptyset$
    let $\mu(\varphi)$ **be** $\emptyset$
    foreach $\theta : A \to \{\mathit{true}, \mathit{false}\}$ **do**
      let e_θ **be** the list of atoms a with $\theta(a) = \mathit{true}$
      let p_θ **be** the proposition $\bigwedge\{a \mid \theta(a) = \mathit{true}\} \wedge \bigwedge\{\neg a \mid \theta(a) = \mathit{false}\}$
      let $\mu^\star(\varphi)$ **be** $\textsc{Merge}([p_\theta \mathrel{?} \varphi\{e^\star_\theta\}], \mu^\star(\varphi))$
      let φ_θ **be** $\varphi\{e_\theta\}$
      if there is $\varphi' \in S$ with $\textsc{Valid}(\varphi_\theta \leftrightarrow \varphi')$
      then let $\mu(\varphi)$ **be** $\textsc{Merge}([p_\theta \mathrel{?} \varphi'], \mu(\varphi))$
      else let S **be** $S \cup \{\varphi_\theta\}$
           let $\mu(\varphi)$ **be** $\textsc{Merge}([p_\theta \mathrel{?} \varphi_\theta], \mu(\varphi))$
           LTL2MT-FSM(φ_θ)
    endfor
    if $\mu(\varphi) = [\mathit{true} \mathrel{?} \varphi]$ and $\mu^\star(\varphi) = [\mathit{true} \mathrel{?} b]$ **then** replace φ by b everywhere
endprocedure

Did we see φ' = φ{θ}?

LTL2MT-FSM algorithm (7)

let S **be** φ
procedure LTL2MT-FSM(φ)
  let $\mu^\star(\varphi)$ **be** $\emptyset$
  let $\mu(\varphi)$ **be** $\emptyset$
  foreach $\theta : A \to \{\mathit{true}, \mathit{false}\}$ **do**
    let e_θ **be** the list of atoms a with $\theta(a) = \mathit{true}$
    let p_θ **be** the proposition $\bigwedge\{a \mid \theta(a) = \mathit{true}\} \wedge \bigwedge\{\neg a \mid \theta(a) = \mathit{false}\}$
    let $\mu^\star(\varphi)$ **be** $\textsc{Merge}([p_\theta \mathbin{?} \varphi\{e^\star_\theta\}], \mu^\star(\varphi))$
    let φ_θ **be** $\varphi\{e_\theta\}$
    if there is $\varphi' \in S$ with $\textsc{Valid}(\varphi_\theta \leftrightarrow \varphi')$
    then $\boxed{\textbf{let } \mu(\varphi) \textbf{ be } \textsc{Merge}([p_\theta \mathbin{?} \varphi'], \mu(\varphi))}$
    else let S **be** $S \cup \{\varphi_\theta\}$
      let $\mu(\varphi)$ **be** $\textsc{Merge}([p_\theta \mathbin{?} \varphi_\theta], \mu(\varphi))$
      LTL2MT-FSM(φ_θ)
  endfor
  if $\mu(\varphi) = [\mathit{true} \mathbin{?} \varphi]$ and $\mu^\star(\varphi) = [\mathit{true} \mathbin{?} b]$ **then** replace φ by b everywhere
endprocedure

Yes: modify the transition set of φ to point to φ'

LTL2MT-FSM algorithm (8)

let S **be** φ

procedure LTL2MT-FSM(φ)

let $\mu^\star(\varphi)$ **be** $\emptyset$

let $\mu(\varphi)$ **be** $\emptyset$

foreach $\theta : A \to \{\mathit{true}, \mathit{false}\}$ **do**

let e_θ **be** the list of atoms a with $\theta(a) = \mathit{true}$

let p_θ **be** the proposition $\bigwedge\{a \mid \theta(a) = \mathit{true}\} \wedge \bigwedge\{\neg a \mid \theta(a) = \mathit{false}\}$

let $\mu^\star(\varphi)$ **be** $\textsc{Merge}([p_\theta \mathbin{?} \varphi\{e^\star_\theta\}], \mu^\star(\varphi))$

let φ_θ **be** $\varphi\{e_\theta\}$

if there is $\varphi' \in S$ with $\textsc{Valid}(\varphi_\theta \leftrightarrow \varphi')$

then let $\mu(\varphi)$ **be** $\textsc{Merge}([p_\theta \mathbin{?} \varphi'], \mu(\varphi))$

else let S **be** $S \cup \{\varphi_\theta\}$

let $\mu(\varphi)$ **be** $\textsc{Merge}([p_\theta \mathbin{?} \varphi_\theta], \mu(\varphi))$

LTL2MT-FSM(φ_θ)

endfor

if $\mu(\varphi) = [\mathit{true} \mathbin{?} \varphi]$ and $\mu^\star(\varphi) = [\mathit{true} \mathbin{?} b]$ **then** repl

endprocedure

No:

1. Add φ' to S,
2. add a non-terminal state to μ(φ),
3. what formulas can φ'(θ) rewrite to?

LTL2MT-FSM algorithm (9)

let S **be** φ
procedure LTL2MT-FSM(φ)
  let $\mu^\star(\varphi)$ **be** $\emptyset$
  let $\mu(\varphi)$ **be** $\emptyset$
  foreach $\theta : A \rightarrow \{\mathit{true}, \mathit{false}\}$ **do**
    let e_θ **be** the list of atoms a with $\theta(a) = \mathit{true}$
    let p_θ **be** the proposition $\bigwedge\{a \mid \theta(a) = \mathit{true}\} \wedge \bigwedge\{\neg a \mid \theta(a) = \mathit{false}\}$
    let $\mu^\star(\varphi)$ **be** $\textsc{Merge}([p_\theta \ ? \ \varphi\{e_\theta^\star\}], \mu^\star(\varphi))$
    let φ_θ **be** $\varphi\{e_\theta\}$
    if there is $\varphi' \in S$ with $\textsc{Valid}(\varphi_\theta \leftrightarrow \varphi')$
    then let $\mu(\varphi)$ **be** $\textsc{Merge}([p_\theta \ ? \ \varphi'], \mu(\varphi))$
    else let S **be** $S \cup \{\varphi_\theta\}$
      let $\mu(\varphi)$ **be** $\textsc{Merge}([p_\theta \ ? \ \varphi_\theta], \mu(\varphi))$
      LTL2MT-FSM(φ_θ)
  endfor
  if $\mu(\varphi) = [\mathit{true} \ ? \ \varphi]$ and $\mu^\star(\varphi) = [\mathit{true} \ ? \ b]$ **then** replace φ by b everywhere
endprocedure

Optimization???

LTL2MT-FSM algorithm (10)

let S **be** φ
procedure LTL2MT-FSM(φ)
  let $\mu^\star(\varphi)$ **be** $\emptyset$
  let $\mu(\varphi)$ **be** $\emptyset$
  foreach $\theta : A \to \{true, false\}$ **do**
    let e_θ **be** the list of atoms a with $\theta(a) = true$
    let p_θ **be** the proposition $\bigwedge\{a \mid \theta(a) = true\} \wedge \bigwedge\{\neg a \mid \theta(a) = false\}$
    let $\mu^\star(\varphi)$ **be** $\textsc{Merge}([p_\theta \; ? \; \varphi\{e_\theta^\star\}], \mu^\star(\varphi))$
    let φ_θ **be** $\varphi\{e_\theta\}$
    if there is $\varphi' \in S$ with $\textsc{Valid}(\varphi_\theta \leftrightarrow \varphi')$
    then let $\mu(\varphi)$ **be** $\textsc{Merge}([p_\theta \; ? \; \varphi'], \mu(\varphi))$
    else let S **be** $S \cup \{\varphi_\theta\}$
      let $\mu(\varphi)$ **be** $\textsc{Merge}([p_\theta \; ? \; \varphi_\theta], \mu(\varphi))$
      LTL2MT-FSM(φ_θ)
  endfor
  if $\mu(\varphi) = [true \, ? \, \varphi]$ and $\mu^\star(\varphi) = [true \, ? \, b]$ **then** replace φ by b everywhere
endprocedure

By now, we generated all possible LTL formulas to which φ can ever evolve (modulo finite state semantics)

By (the almighty) theorem 2, this will terminate

LTL2MT-FSM algorithm (6, again)

let S **be** φ
procedure LTL2MT-FSM(φ)
  let $\mu^\star(\varphi)$ **be** $\emptyset$
  let $\mu(\varphi)$ **be** $\emptyset$
  foreach $\theta : A \to \{\mathit{true}, \mathit{false}\}$ **do**
    let e_θ **be** the list of atoms a with $\theta(a) = \mathit{true}$
    let p_θ **be** the proposition $\bigwedge\{a \mid \theta(a) = \mathit{true}\} \wedge \bigwedge\{\neg a \mid \theta(a) = \mathit{false}\}$
    let $\mu^\star(\varphi)$ **be** $\textsc{Merge}([p_\theta \ ? \ \varphi\{e^\star_\theta\}], \mu^\star(\varphi))$
    let φ_θ **be** $\varphi\{e_\theta\}$
    if there is $\varphi' \in S$ with $\textsc{Valid}(\varphi_\theta \leftrightarrow \varphi')$
    then let $\mu(\varphi)$ **be** $\textsc{Merge}([p_\theta \ ? \ \varphi'], \mu(\varphi))$
    else let S **be** $S \cup \{\varphi_\theta\}$
      let $\mu(\varphi)$ **be** $\textsc{Merge}([p_\theta \ ? \ \varphi_\theta], \mu(\varphi))$
      LTL2MT-FSM(φ_θ)
  endfor
  if $\mu(\varphi) = [\mathit{true} \ ? \ \varphi]$ and $\mu^\star(\varphi) = [\mathit{true} \ ? \ b]$ **then** replace φ by b everywhere
endprocedure

Homework: what is valid?

What we saw in this lecture…

- LTL syntax and semantics
- Intro to BDDs
- “Special” FSMs that LTL specs get translated to
- Algorithms for translating LTL to FSMs

Next lecture

- ERE monitor synthesis
  - Reading is assigned