Runtime Verification
From theory to practice and back

Gordon J. Pace
(joint course design with Christian Colombo)
University of Malta

September 2016

Part I

LTL

LTL Formulae

- Example of specification of good behaviour (assuming events don't happen concurrently):
  - `!login Until initialise`

LTL

$$
\begin{array}{lll}
\psi & ::= & \text{event} \\
 & | & \neg\psi \\
 & | & \psi \wedge \psi' \\
 & | & \mathsf{X}\ \psi \\
 & | & \mathsf{G}\ \psi \\
 & | & \psi\ \mathsf{U}\ \psi'
\end{array}
$$

LTL

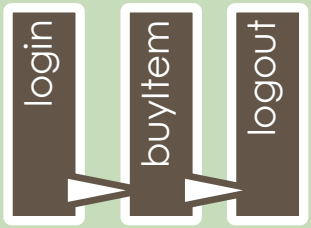
$$
\begin{array}{lll}
\psi & ::= & \text{event} \\
 & | & \neg\,\psi \\
 & | & \psi \wedge \psi' \\
 & | & X\ \psi \\
 & | & G\ \psi \\
 & | & \psi\ U\ \psi'
\end{array}
$$

This is the first event of the behaviour of the system

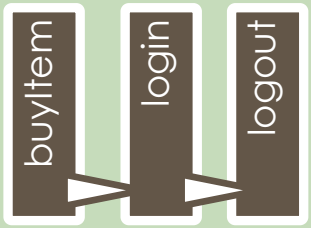
LTL

$$
\begin{array}{lll}
\psi & ::= & \text{event} \\
 & | & \neg\,\psi \\
 & | & \psi \wedge \psi' \\
 & | & X\ \psi \\
 & | & G\ \psi \\
 & | & \psi\ U\ \psi'
\end{array}
$$

The formula "login" is satisfied by the trace:



but not by:



LTL

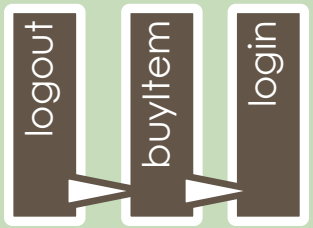
$$
\begin{array}{lll}
\psi & ::= & \text{event} \\
 & | & \neg\psi \\
 & | & \psi \wedge \psi' \\
 & | & X\ \psi \\
 & | & G\ \psi \\
 & | & \psi\ U\ \psi'
\end{array}
$$

Formula ψ does not hold

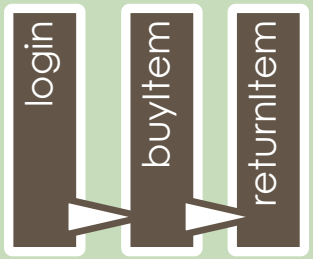
LTL

$$
\begin{array}{lll}
\psi & ::= & \text{event} \\
 & | & \neg\,\psi \\
 & | & \psi \wedge \psi' \\
 & | & X\ \psi \\
 & | & G\ \psi \\
 & | & \psi\ U\ \psi'
\end{array}
$$

Formula "¬login" is satisfied by the trace:



but not by:



LTL

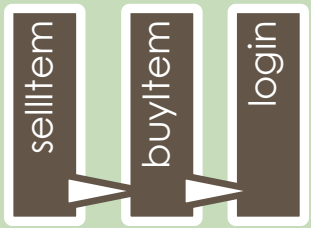
$$
\begin{array}{lll}
\psi & ::= & \text{event} \\
 & | & \neg\psi \\
 & | & \psi \wedge \psi' \\
 & | & X\ \psi \\
 & | & G\ \psi \\
 & | & \psi\ U\ \psi'
\end{array}
$$

Both ψ and ψ' hold

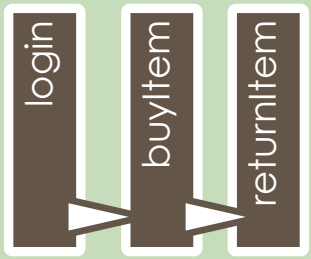
LTL

$$
\begin{array}{lll}
\psi & ::= & \text{event} \\
 & | & \neg\,\psi \\
 & | & \psi \wedge \psi' \\
 & | & X\ \psi \\
 & | & G\ \psi \\
 & | & \psi\ U\ \psi'
\end{array}
$$

Formula "¬login ∧ ¬logout" is satisfied by:



but not by:



LTL

$$
\begin{array}{lll}
\psi & ::= & \text{event} \\
 & | & \neg\psi \\
 & | & \psi \wedge \psi' \\
 & | & X\ \psi \\
 & | & G\ \psi \\
 & | & \psi\ U\ \psi'
\end{array}
$$

Other logic operators may be defined as syntactic sugar of these two:

$$\psi \vee \psi' \equiv \neg(\neg\psi \wedge \neg\psi')$$
$$\psi \Rightarrow \psi' \equiv \neg\psi \vee \psi'$$
$$\psi \Leftrightarrow \psi' \equiv$$
$$\psi \Rightarrow \psi' \wedge \psi' \Rightarrow \psi$$

LTL

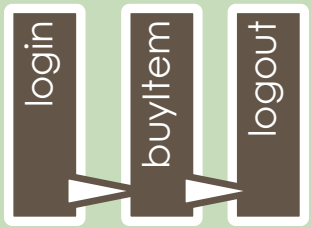
$$
\begin{array}{lll}
\psi & ::= & \text{event} \\
 & | & \neg\,\psi \\
 & | & \psi \wedge \psi' \\
 & | & X\ \psi \\
 & | & G\ \psi \\
 & | & \psi\ U\ \psi'
\end{array}
$$

Ignoring the current event, ψ will hold

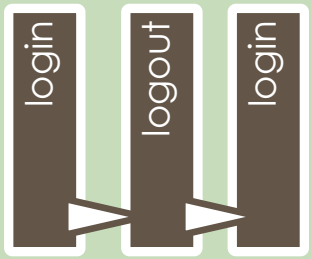
LTL

$$
\begin{array}{lll}
\psi & ::= & \text{event} \\
 & | & \neg\,\psi \\
 & | & \psi \wedge \psi' \\
 & | & X\ \psi \\
 & | & G\ \psi \\
 & | & \psi\ U\ \psi'
\end{array}
$$

Formula "login ∧ ¬X logout" is satisfied by:



but not by:



LTL

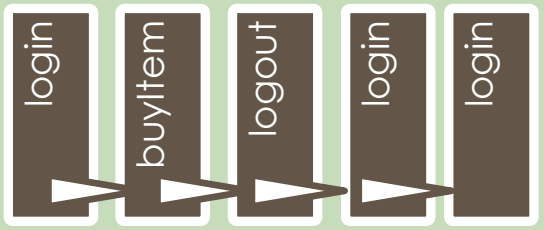
$$
\begin{array}{lll}
\psi & ::= & \text{event} \\
 & | & \neg\psi \\
 & | & \psi \wedge \psi' \\
 & | & X\ \psi \\
 & | & G\ \psi \\
 & | & \psi\ U\ \psi'
\end{array}
$$

Formula ψ will hold in all future points in time

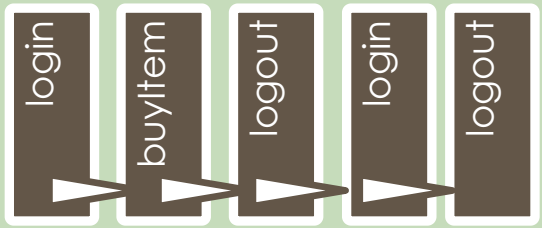
LTL

$$
\begin{array}{lll}
\psi & ::= & \text{event} \\
 & | & \neg\,\psi \\
 & | & \psi \wedge \psi' \\
 & | & X\ \psi \\
 & | & G\ \psi \\
 & | & \psi\ U\ \psi'
\end{array}
$$

"G (login⇒¬X logout)"
is satisfied by:



but not by:



LTL

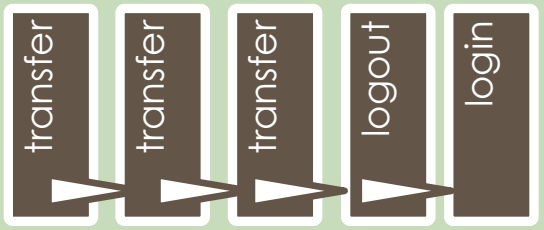
$$
\begin{array}{lll}
\psi & ::= & \text{event} \\
 & | & \neg\psi \\
 & | & \psi \wedge \psi' \\
 & | & \mathsf{X}\ \psi \\
 & | & \mathsf{G}\ \psi \\
 & | & \psi\ \mathsf{U}\ \psi'
\end{array}
$$

Eventually, formula ψ' will hold, but until then ψ will hold

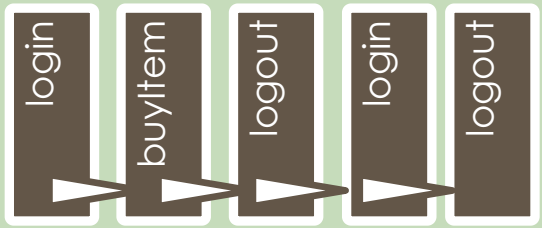
LTL

$$
\begin{array}{lll}
\psi & ::= & \text{event} \\
 & | & \neg\,\psi \\
 & | & \psi \wedge \psi' \\
 & | & X\ \psi \\
 & | & G\ \psi \\
 & | & \psi\ U\ \psi'
\end{array}
$$

"transfer U X logout"
is satisfied by:



but not by:



LTL

$$
\begin{array}{lll}
\psi & ::= & \text{event} \\
 & | & \neg\psi \\
 & | & \psi \wedge \psi' \\
 & | & X\ \psi \\
 & | & G\ \psi \\
 & | & \psi\ U\ \psi'
\end{array}
$$

Other temporal operators may be defined as syntactic sugar as with:

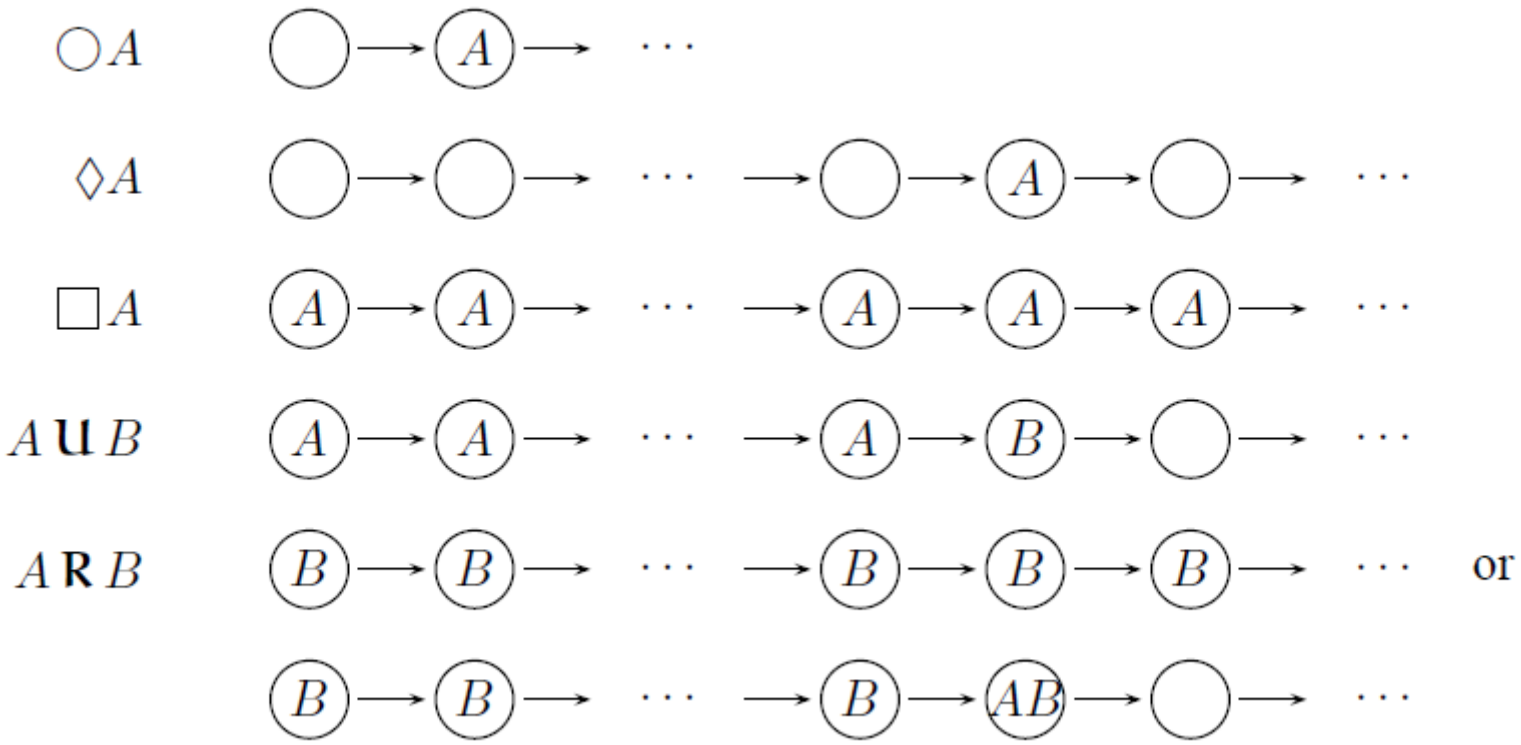
$F\psi \equiv \text{true}\ U\ \psi$

$\psi\ R\ \psi' = \neg(\neg\ \psi\ U \neg\ \psi')$

Even the G operator could have been defined:

$G\psi \equiv \neg(F\neg\psi)$

Pictorially…



LTL

- A logout must have been preceded by a login:
  $\neg \text{logout}\ U\ \text{login}$
- More generally, for multiple sessions:
  $(\neg \text{logout}\ U\ \text{login})$
  $\wedge\ (G(\text{logout} \Rightarrow X\ (\neg \text{logout}\ U\ \text{login})))$
- Transfers only occur between a login and a logout (single session version):
  $(\neg \text{transfer}\ U\ \text{login})$
  $\wedge\ (G\ (\text{logout} \Rightarrow G \neg \text{transfer}))$

LTL Formulae

- Example of specification of good behaviour:
  - `!login Until initialise`
  - `(!transfer Until login) and (Globally (logout => Next (!transfer Until login)))`

Exercise

- Express properties 2, 5, and 10 of the Financial Transaction System in terms of LTL

Example

Property 2 - The transaction system must be initialised before any user logs in.

```
property matching {

      (not USER_login) Until initialise

 }
```

More Advanced Example

Property 5 – Once a user is disabled, he or she may not withdraw from an account until being made activate again.

```
property foreach target (UserInfo u) matching {
  ((not transfer) Until login) and (Globally
(logout => Next((not transfer) Until login)))
}
```

Monitoring LTL Formulae

- Verification can be done either by:
  - By using the residual (rewriting) algorithm or
  - Transforming into an automaton

Disadvantage of Residuals

- While using residuals it is quite easy to build a monitor…
- … it involves significant overhead at runtime

(imagine how many times the rules would have to be checked for a significantly long LTL formula)

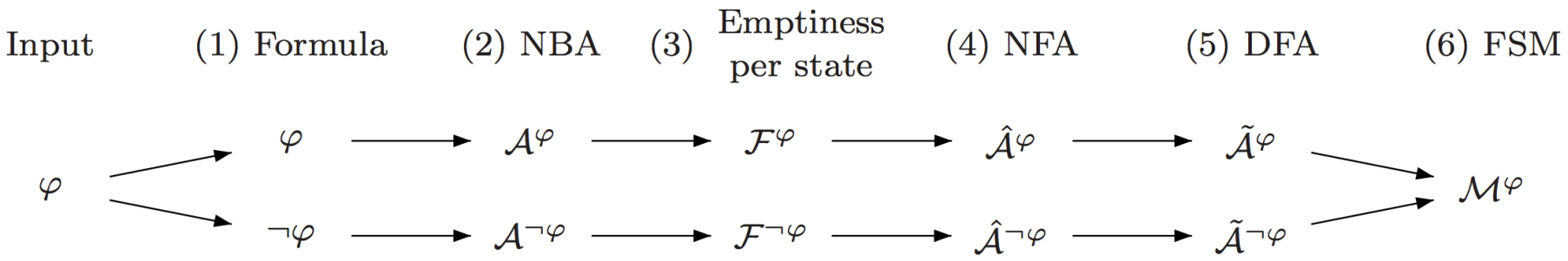
Part II

Monitoring LTL using Automata

LTL – yes, no, ?

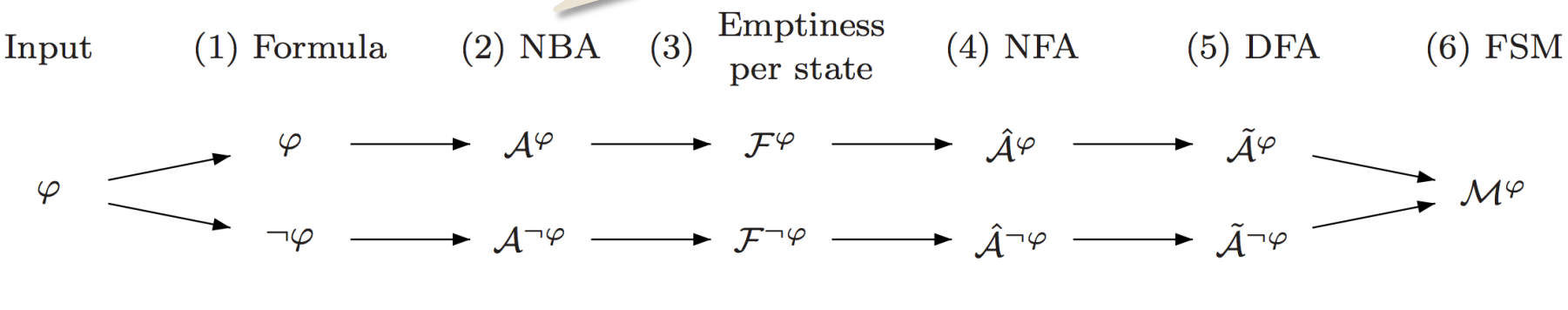
$$[u \models \varphi] = \begin{cases} \top & \text{if } \forall \sigma \in \Sigma^{\omega} : u\sigma \models \varphi \\ \bot & \text{if } \forall \sigma \in \Sigma^{\omega} : u\sigma \not\models \varphi \\ ? & \text{otherwise} \end{cases}$$

LTL to Automata



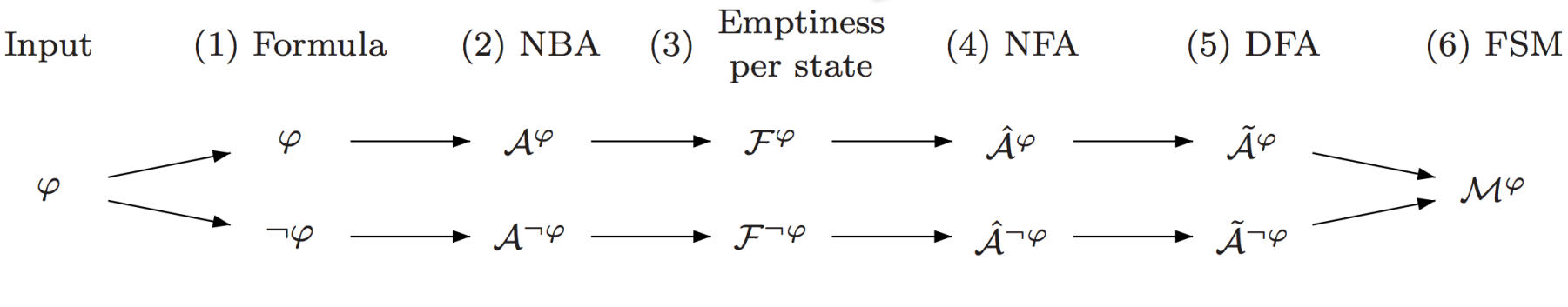
LTL to Automata

Standard translation to Buchi automata



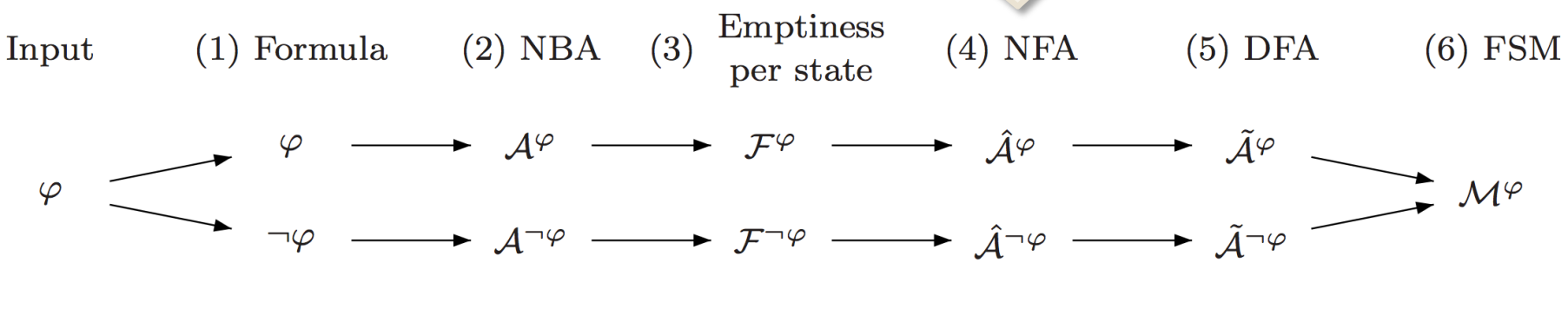
LTL to Automata

Identify states which can lead to violation



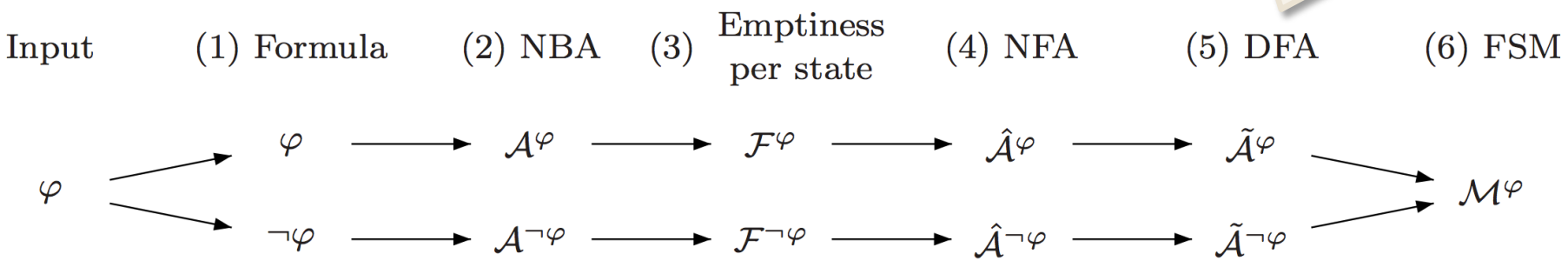
LTL to Automata

Generate finite automaton whose final states correspond to states leading to violation

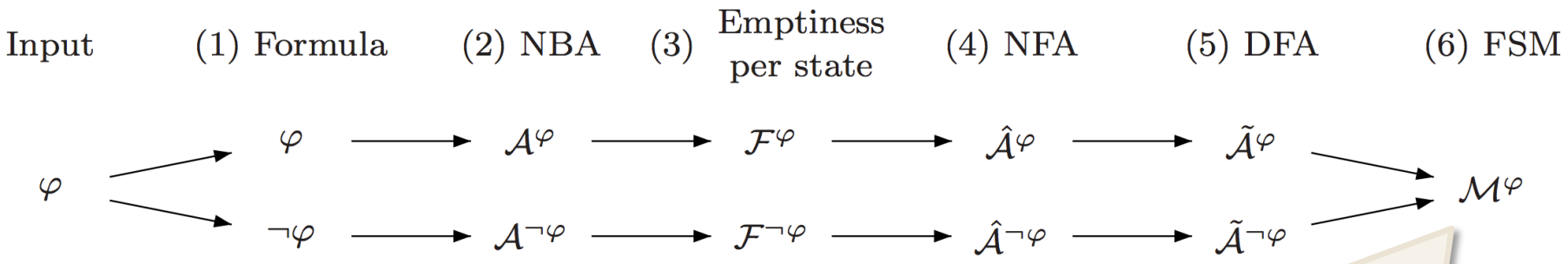


LTL to Automata

Determinise the automaton



LTL to Automata



Combine both decision procedures to achieve:

$$[u \models \varphi] = \begin{cases} \top & \text{if } \forall \sigma \in \Sigma^{\omega} : u\sigma \models \varphi \\ \bot & \text{if } \forall \sigma \in \Sigma^{\omega} : u\sigma \not\models \varphi \\ ? & \text{otherwise} \end{cases}$$

Exercises

- Edit the method *toAspectJAutomata( )* in LTLExp.java to generate the product automaton.