## Model Checking

Tom Henzinger IST Austria Specification Omega Automata

Syntax as for finite automata, in addition one of the following acceptance conditions:

Buchi:	$BA \subseteq S$
coBuchi:	$CA \subseteq S$
Streett:	$SA \subseteq 2^S \times 2^S$
Rabin:	$RA \subseteq 2^{S} \times 2^{S}$

Language L(M) of specification omega-automaton  

$$M = (S, S_0, \rightarrow, \phi, A):$$

infinite trace 
$$t_0, t_1, ... \in L(M)$$
  
iff

there exists an infinite run  $s_0 \rightarrow s_1 \rightarrow ...$  of M such that 1.  $s_0 \rightarrow s_1 \rightarrow ...$  satisfies A

2. for all  $i \ge 0$ ,  $t_i \models \phi(s_i)$ 

Let  $Inf(s) = \{ p | p = s_i \text{ for infinitely many } i \}.$ 

The infinite run s satisfies the acceptance condition A iff

- Buchi:  $Inf(s) \cap BA \neq \emptyset$
- coBuchi:  $Inf(s) \subseteq CA$
- Streett: for all  $(l,r) \in SA$ , if  $Inf(s) \cap l \neq \emptyset$  then  $Inf(s) \cap r \neq \emptyset$
- Rabin:for some  $(l,r) \in RA$ , $Inf(s) \cap l = \emptyset$  and  $Inf(s) \cap r \neq \emptyset$

finite:	$\diamond$ FA
Buchi:	$\Box \diamondsuit BA$
coBuchi:	$\bigcirc \Box CA$
Streett:	$\wedge (\Box \Diamond I \Rightarrow \Box \Diamond r)$
Rabin:	∨ (�□¬l ∧ □�r)

### Linear semantics of specification omega automata: omega-language containment



Response specification automaton :  $\Box$  (a  $\Rightarrow$   $\diamond$ b) assuming (a  $\land$  b) = false



Buchi condition  $\{s_0, s_3\}$ 

Linear semantics of monitor omega automata: omega-language emptiness

 $(K,q) = M \quad \text{iff} \quad L(K,q) \land L(M) = ;$ 

Response monitor automaton :  $\Box$  (a  $\Rightarrow$   $\diamond$ b) assuming (a  $\land$  b) = false



Buchi condition  $\{s_2\}$ 



Buchi condition  $\{s_0\}$ No coBuchi condition Streett condition {  $({s_0, s_1}, {s_0})$  } Rabin condition {  $(\emptyset, {s_0})$  }



No Buchi conditionStreett condition  $\{ (\{s_1\}, \emptyset) \}$ coBuchi condition  $\{ s_0 \}$ Rabin condition  $\{ (\{s_1\}, \{s_0, s_1\}) \}$ 





Buchi condition  $\{s_2\}$ 

-Buchi and coBuchi automata cannot be determinized -Streett and Rabin automata can be determinized

nondeterministic Buchi =

deterministic Streett = deterministic Rabin =

nondeterministic Streett = nondeterministic Rabin = omega-regular Omega automata are strictly more expressive than LTL.

Omega-automata: omega-regular languages U LTL: counter-free omega-regular languages

#### Omega automata: omega-regular languages

- = second-order theory of monadic predicates & successor
- = omega-regular expressions

## J

- LTL: counter-free omega-regular languages
- = first-order theory of monadic predicates & successor
- = star-free omega-regular expressions



$$(\forall p) (p \land \bigcirc \neg p \land \square (p \Leftrightarrow \bigcirc \bigcirc p) \Rightarrow \square (p \Rightarrow a))$$

 $\begin{array}{l} (\forall p) ( p(0) \land \neg p(1) \land (\forall t) (p(t) \Leftrightarrow p(t+2)) \Rightarrow \\ (\forall t) (p(t) \Rightarrow a(t))) \end{array}$ 

(a; true)<sup>ω</sup>

Structure of the Omega-Regular Languages



Structure of the Omega-Regular Languages



counter-free

#### Structure of the Counter-free Omega-Regular Languages



The location of a linear-time property in the Borel hierarchy indicates how hard (theoretically as well as conceptually) the corresponding model-checking problem is.



Model-checking problem



Model-checking problem



easiest harder hard Model-Checking Algorithms = Graph Algorithms

1 Safety:

-solve: ∃U model checking, finite monitors (◇ emptiness) -algorithm: reachability (linear)

2 Eventuality under weak fairness:

-solve: weakly fair CTL ( $\exists \Box$  model checking), Buchi monitors ( $\Box \diamondsuit$  emptiness)

-algorithm: strongly connected components (linear)

3 Liveness:

-solve: strongly fair CTL, Streett monitors (∧ (◇□∨□◇) emptiness)
-algorithm: recursively nested SCCs (quadratic) From specification automata to monitor automata: determinization (exponential) + complementation (easy)

From LTL to monitor automata: complementation (easy) + tableau construction (exponential) Five Key Algorithms

- 1 Reachability
- 2 Strongly connected components
- 3 Recursively nested SCCs
- 4 Tableau construction
- 5 Streett determinization

#### Finite Emptiness

- Given: finite automaton (S, S<sub>0</sub>,  $\rightarrow$ ,  $\phi$ , FA)
- Find: is there a path from a state in  $S_0$  to a state in FA?

Solution: depth-first or breadth-first search

Application 1:9U model checkingApplication 2:finite monitors

#### **Buchi Emptiness**

Given: Buchi automaton (S,  $S_0$ ,  $\rightarrow$ ,  $\phi$ , BA)

Find: is there an infinite path from a state in  $S_0$  that visits some state in BA infinitely often?

- Solution: 1. Compute SCC graph by depth-first search
  - 2. Mark SCC C as fair iff  $C \cap BA \neq \emptyset$
  - 3. Check if some fair SCC is reachable from  $S_0$

Application 1:	CTL model checking over weakly-fair transition graphs
	(note: really need <mark>multiBuchi</mark> )
Application 2:	Buchi monitors

#### Streett Emptiness

Given: Streett automaton (S,  $S_0$ ,  $\rightarrow$ ,  $\phi$ , SA)

Find: is there an infinite path from a state in  $S_0$  that satisfies all Streett conditions (I,r) in SA?

Solution: check if  $S_0 \cap \text{RecSCC}(S, \rightarrow, SA) \neq \emptyset$ 

```
function RecSCC (S, \rightarrow, SA):
         X := \emptyset
         for each C \in SCC(S, \rightarrow) do
                   F := \emptyset
                  if \rightarrow_{c} \neq \emptyset then
                            for each (l,r) \in SA do
                                      if C \cap r \neq \emptyset
                                               then F := F \cup (l,r)
                                               else C := C \setminus I
                            if F = SA
                                      then X := X \cup pre<sup>*</sup>(C)
                                      else X := X \cup RecSCC (C, \rightarrow_{c}, F)
         return X
```

#### Complexity

- n number of states m number of transitions
- s number of Streett pairs

Reachability:	O(n+m)
SCC:	O(n+m)
RecSCC:	O((n+m) · s²)

Application 1:	CTL model checking over strongly-fair transition graphs
Application 2:	Streett monitors

#### Tableau Construction

Given: LTL formula  $\varphi$ Find: Buchi automaton  $M_{\varphi}$  such that  $L(M_{\varphi}) = L(\varphi)$ monitors subformulas of  $\varphi$ 

#### Fischer-Ladner Closure of a Formula

Sub (a)={ a }Sub ( $\phi \land \psi$ )={  $\phi \land \psi$  }  $\cup$  Sub ( $\phi$ )  $\cup$  Sub ( $\psi$ )Sub ( $\neg \phi$ )={  $\neg \phi$  }  $\cup$  Sub ( $\phi$ )Sub ( $\bigcirc \phi$ )={  $\bigcirc \phi$  }  $\cup$  Sub ( $\phi$ )Sub ( $\phi \cup \psi$ )={  $\phi \cup \psi$ ,  $\bigcirc (\phi \cup \psi)$  }  $\cup$  Sub ( $\phi$ )  $\cup$  Sub ( $\psi$ )

| Sub ( $\varphi$ ) | = O( $|\varphi|$ )

# $s \subseteq Sub(\phi)$ is consistent iff

-if  $(\psi \land \chi) \in \text{Sub}(\varphi)$  then  $(\psi \land \chi) \in \text{s}$  iff  $\psi \in \text{s}$  and  $\chi \in \text{s}$ -if  $(\neg \psi) \in \text{Sub}(\varphi)$  then  $(\neg \psi) \in \text{s}$  iff  $\psi \notin \text{s}$ -if  $(\psi U \chi) \in \text{Sub}(\varphi)$  then  $(\psi U \chi) \in \text{s}$  iff either  $\chi \in \text{s}$ or  $\psi \in \text{s}$  and  $\bigcirc (\psi U \chi) \in \text{s}$ 

Tableau 
$$M_{\phi} = (S, S_0, \rightarrow, \phi, BA)$$

S ... set of consistent subsets of Sub ( $\varphi$ )

$$\textbf{s} \in \textbf{S}_0 \text{ iff } \phi \in \textbf{s}$$

$$\begin{array}{ll} \mathsf{s} \to \mathsf{t} & \text{iff for all } (\bigcirc \psi) \in \ \mathsf{Sub} \ (\phi), \\ & (\bigcirc \psi) \in \ \mathsf{s} & \text{iff } \ \psi \in \ \mathsf{t} \end{array} \end{array}$$

 $\phi(s)$  ... conjunction of atomic observations in s and negated atomic observations not in s

For each  $(\psi U \chi) \in \text{Sub}(\varphi)$ , BA contains { s |  $\chi \in$  s or  $(\psi U \chi) \notin$  s }













#### Size of $M_{\phi}$ is $O(2^{|\phi|})$ .

CTL model checking:linear / quadraticLTL model checking:PSPACE-complete

#### Graph Algorithms

Given: labeled graph  $(Q, \rightarrow, A, [])$ Cost: each node access and edge access has unit cost Complexity: in terms of  $|Q| = n \dots$  number of nodes  $|\rightarrow| = m \dots$  number of edges Reachability and s.c.c.s: O(m+n) The Graph-Algorithmic View is Problematic

-The graph is given implicitly (by a program) not explicitly (e.g., by adjacency lists).

-Building an explicit graph representation is exponential, but usually unnecessary ("on-the-fly" algorithms).

-The explicit graph representation may be so big, that the "unit-cost model" is not realistic.

-A class of algorithms, called "symbolic algorithms", do not operate on nodes and edges at all.

#### Symbolic Model-Checking Algorithms

Given: a "symbolic theory", that is, an abstract data type called region with the following operations:

pre,  $\forall \texttt{pre, post}, \forall \texttt{post}: \texttt{region} \rightarrow \texttt{region}$ 

 $\cap, \cup, \setminus: \text{ region} \times \text{region} \rightarrow \text{region}$ 

 $\subseteq \text{, =}: \text{ region} \times \text{region} \rightarrow \text{bool}$ 

 $\langle \rangle$ ,  $\rangle \langle : A \rightarrow region$ 

 $\emptyset, \mathbf{Q}: region$ 

#### Intended Meaning of Symbolic Theories

set of states region •••  $\cap, \cup, \setminus, \subseteq, =, \emptyset$  ... set operations  $\langle a \rangle = \{ q \in Q \mid [q] = a \}$  $a < = \{ q \in \mathbb{Q} \mid [q] \neq a \}$ pre (R) = {  $q \in Q \mid (\exists r \in R) q \rightarrow r$  }  $\forall \mathsf{pre}(\mathsf{R}) = \{ q \in \mathsf{Q} \mid (\forall r)(q \rightarrow r \Rightarrow r \in \mathsf{R}) \}$ post (R) = {  $q \in Q \mid (\exists r \in R) r \rightarrow q$  }  $\forall post(R) = \{ q \in Q \mid (\forall r)(r \rightarrow q \Rightarrow r \in R) \}$  If the state of a system is given by variables of type Vals, and the transitions of the system can be described by operations Ops on Vals, then the first-order theory FO (Vals, Ops) is an adequate symbolic theory:

region...formula of FO (Vals, Ops) $\cap, \cup, \setminus, \subseteq, =, \emptyset, Q$  $\wedge, \vee, , \Rightarrow$  validity,  $\Leftrightarrow$  validity, f, tpre (R(X))= (\exists X')( Trans(X,X')  $\wedge$  R(X') ) $\forall$ pre (R(X))= ( $\forall$  X')( Trans(X,X')  $\Rightarrow$  R(X') )post (R(X))= ( $\exists$  X")( R(X")  $\wedge$  Trans(X",X) ) $\forall$ post (R(X))= ( $\forall$  X")( Trans(X",X)  $\Rightarrow$  R(X") )

If FO (Vals, Ops) admits quantifier elimination, then the propositional theory ZO (Vals, Ops) is an adequate symbolic theory:

each pre/post operation is a quantifier elimination

Example: Boolean Systems

-all system variables X are boolean -region: quantifier-free boolean formula over X -pre, post: boolean quantifier elimination

Complexity: PSPACE

Example: Presburger Systems

-all system variables X are integers

-the transition relation  $\mbox{Trans}(X,X')$  is defined using only  $\leq$  and +

-region: quantifier-free formula of  $(Z, \leq, +)$ 

-pre, post: quantifier elimination

An iterative language for writing symbolic model-checking algorithms

-only data type is region -expressions: pre, post,  $\cap$ ,  $\cup$ ,  $\setminus$ ,  $\subseteq$ , =, < >,  $\emptyset$ , Q -assignment, sequencing, while-do, if-then-else Example: Reachability  $\exists \diamondsuit a$ 

