Verification – Lecture 15
Computation Tree Logic

Bernd Finkbeiner – Sven Schewe
Rayna Dimitrova – Lars Kuhtz – Anne Proetzsch

Wintersemester 2007/2008

REVIEW

Summary of LTL model checking (1)

- LTL is a logic for formalizing path-based properties
- Expansion law allows for rewriting until into local conditions and next
- LTL-formula φ can be transformed algorithmically into NBA $\mathcal{A}_\varphi$
  - this may cause an exponential blow up
  - algorithm: first construct a GNBA for φ; then transform it into an equivalent NBA
- LTL-formulae describe ω-regular LT-properties
  - but do not have the same expressivity as ω-regular languages

Summary of LTL model checking (2)

- $S \models \varphi$ can be solved by a nested depth-first search in $S \otimes \mathcal{A}_{\neg\varphi}$
  - time complexity of the LTL model-checking algorithm is linear in S and exponential in $|\varphi|$
- Fairness assumptions can be described by LTL-formulae

  the model-checking problem for LTL with fairness is reducible to the standard LTL model-checking problem
- The LTL-model checking problem is PSPACE-complete
- Satisfiability and validity of LTL amounts to NBA emptiness-check
- The satisfiability and valditiy problem for LTL are PSPACE-complete

Linear and branching temporal logic

- *Linear* temporal logic:

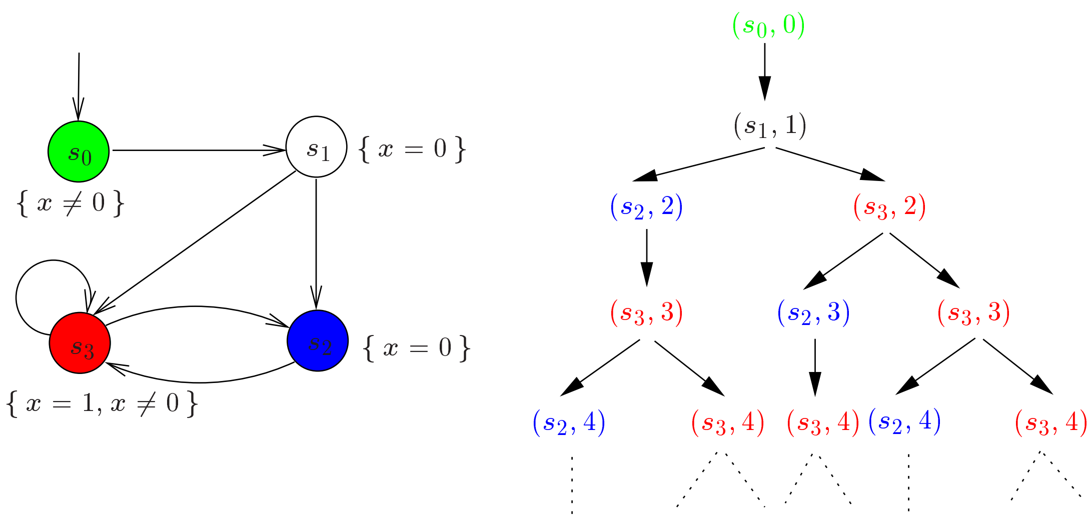
  "statements about (all) paths starting in a state"
  - $s \models \Box (x \leqslant 20)$ iff for all possible paths starting in s always $x \leqslant 20$
- *Branching* temporal logic:

  "statements about all or some paths starting in a state"
  - $s \models \forall\Box (x \leqslant 20)$ iff for **all** paths starting in s always $x \leqslant 20$
  - $s \models \exists\Box (x \leqslant 20)$ iff for **some** path starting in s always $x \leqslant 20$
  - nesting of path quantifiers is allowed
- Checking $\exists\varphi$ in LTL can be done using $\forall\neg\varphi$
  - . . . but this does not work for nested formulas such as $\forall\Box\,\exists\Diamond\, a$

Linear versus branching temporal logic

- Semantics is based on a branching notion of time
  - an infinite tree of states obtained by unfolding state graph
  - one “time instant” may have several possible successor “time instants”
- Incomparable expressiveness
  - there are properties that can be expressed in LTL, but not in CTL
  - there are properties that can be expressed in most branching, but not in LTL
- Distinct model-checking algorithms, and their time complexities
- Distinct treatment of fairness assumptions
- Distinct equivalences (pre-orders) on state graphs
  - that correspond to logical equivalence in LTL and branching temporal logics

State graphs and trees

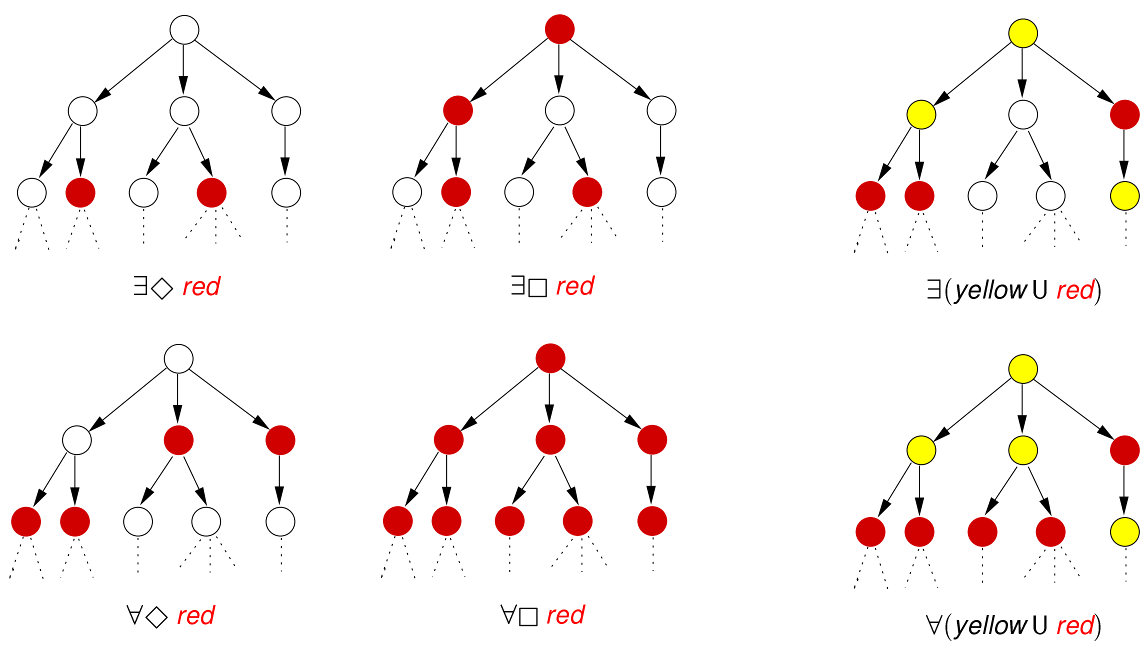


Branching temporal logics

There are various branching temporal logics:

- Hennessy-Milner logic
- Computation Tree Logic (CTL)
- Extended Computation Tree Logic (CTL*)
  - combines LTL and CTL into a single framework
- Alternation-free modal μ-calculus
- Modal μ-calculus
- Propositional dynamic logic

Computation tree logic (CTL)



“behavior” in a state s	path-based: set of paths starting in s	state-based: computation tree of s
temporal logic	LTL: path formulas φ $s \models \varphi$ iff $\forall \pi \in \mathit{Paths}(s).\, \pi \models \varphi$	CTL: state formulas existential path quantification $\exists\varphi$ universal path quantification: $\forall\varphi$
complexity of the model checking problems	PSPACE–complete $\mathcal{O}\left(\lvert S\rvert \cdot 2^{\lvert\varphi\rvert}\right)$	PTIME $\mathcal{O}\left(\lvert S\rvert \cdot \lvert\Phi\rvert\right)$
implementation-relation	trace inclusion and the like (proof is PSPACE-complete)	simulation and bisimulation (proof in polynomial time)
fairness	no special techniques	special techniques needed

Syntax

modal logic over infinite trees [Clarke & Emerson 1981]

- State formulas: $\Phi ::= \text{true} \mid a \mid \Phi_1 \wedge \Phi_2 \mid \neg\Phi \mid \exists\varphi \mid \forall\varphi$
  - $a \in AP$ — atomic proposition
  - $\neg\,\Phi$ and $\Phi_1 \wedge \Phi_2$ — negation and conjunction
  - $\exists\varphi$ — there *exists* a path fulfilling φ
  - $\forall\varphi$ — *all* paths fulfill φ
- Path formulas: $\varphi :: \bigcirc\Phi \mid \Phi_1 \,\mathsf{U}\, \Phi_2$
  - $\bigcirc\Phi$ — the next state fulfills Φ
  - $\Phi_1 \,\mathsf{U}\, \Phi_2$ — Φ_1 holds until a Φ_2-state is reached

⇒ note that $\bigcirc$ and U *alternate* with $\forall$ and $\exists$

- $\forall\bigcirc\bigcirc\,\Phi$ and $\forall\exists\bigcirc\,\Phi \notin$ CTL, but $\forall\bigcirc\forall\bigcirc\,\Phi$ and $\forall\bigcirc\exists\bigcirc\,\Phi \in$ CTL

Derived operators

potentially Φ:	$\exists\Diamond\Phi$	$=$	$\exists(\text{true}\,\mathsf{U}\,\Phi)$
inevitably Φ:	$\forall\Diamond\Phi$	$=$	$\forall(\text{true}\,\mathsf{U}\,\Phi)$
potentially always Φ:	$\exists\Box\Phi$	$:=$	$\neg\forall\Diamond\neg\Phi$
invariantly Φ:	$\forall\Box\Phi$	$=$	$\neg\exists\Diamond\neg\Phi$
weak until:	$\exists(\Phi\,\mathsf{W}\,\Psi)$	$=$	$\neg\forall\big((\Phi\wedge\neg\Psi)\,\mathsf{U}\,(\neg\Phi\wedge\neg\Psi)\big)$
	$\forall(\Phi\,\mathsf{W}\,\Psi)$	$=$	$\neg\exists\big((\Phi\wedge\neg\Psi)\,\mathsf{U}\,(\neg\Phi\wedge\neg\Psi)\big)$

the boolean connectives are derived as usual

Semantics of CTL state-formulas

Defined by a relation $\models$ such that

$q \models \Phi$ if and only if formula Φ holds in state q

$q \models a$	iff	$a \in L(q)$
$q \models \neg\,\Phi$	iff	$\neg\,(q \models \Phi)$
$q \models \Phi \wedge \Psi$	iff	$(q \models \Phi) \wedge (q \models \Psi)$
$q \models \exists\varphi$	iff	$\pi \models \varphi$ for *some* path $\pi \in$ *Paths*(q)
$q \models \forall\varphi$	iff	$\pi \models \varphi$ for *all* paths $\pi \in$ *Paths*(q)

Notation: *Paths*(q): set of paths starting in q

Semantics of CTL path-formulas

Define a relation $\models$ such that

$\pi \models \varphi$ if and only if path π satisfies φ

$$\pi \models \bigcirc\Phi \quad \text{iff } \pi[1] \models \Phi$$

$$\pi \models \Phi \,\mathsf{U}\, \Psi \quad \text{iff } (\exists\, j \geqslant 0.\, \pi[j] \models \Psi \;\wedge\; (\forall\, 0 \leqslant k < j.\, \pi[k] \models \Phi))$$

where $\pi[i]$ denotes the state q_i in the path $\pi = q_0\, q_1\, q_2 \ldots$

Transition system semantics

- For CTL-state-formula Φ, the *satisfaction set* *Sat*(Φ) is defined by:

$$\mathit{Sat}(\Phi) = \{\, q \in Q \mid q \models \Phi \,\}$$

- State graph *S* satisfies CTL-formula Φ iff Φ holds in all its initial states:

$$S \models \Phi \quad \text{if and only if} \quad \forall q_0 \in Q_0.\, q_0 \models \Phi$$

  - this is equivalent to $Q_0 \subseteq \mathit{Sat}(\Phi)$

- Point of attention: $S \not\models \Phi$ and $S \not\models \neg\Phi$ is possible!
  - because of several initial states, e.g. $q_0 \models \exists\Box\,\Phi$ and $q_0' \not\models \exists\Box\,\Phi$

CTL equivalence

CTL-formulas Φ and Ψ (over *AP*) are *equivalent*, denoted $\Phi \equiv \Psi$ if and only if $\mathit{Sat}(\Phi) = \mathit{Sat}(\Psi)$ for all state graphs *S* over *AP*

$$\Phi \equiv \Psi \quad \text{iff} \quad (S \models \Phi \quad \text{if and only if} \quad S \models \Psi)$$

Duality laws

$$\forall \bigcirc \Phi \equiv \neg \exists \bigcirc \neg \Phi$$

$$\exists \bigcirc \Phi \equiv \neg \forall \bigcirc \neg \Phi$$

$$\forall \Diamond \Phi \equiv \neg \exists \Box \neg \Phi$$

$$\exists \Diamond \Phi \equiv \neg \forall \Box \neg \Phi$$

$$\forall (\Phi \, \mathsf{U} \, \Psi) \equiv \neg \exists ((\Phi \wedge \neg \Psi) \, \mathsf{W} \, (\neg \Phi \wedge \neg \Psi))$$

Expansion laws

Recall in LTL: $\varphi \, \mathsf{U} \, \psi \;\equiv\; \psi \;\vee\; (\varphi \wedge \bigcirc(\varphi \, \mathsf{U} \, \psi))$

In CTL:

$$\begin{aligned}
\forall(\Phi \, \mathsf{U} \, \Psi) &\equiv \Psi \;\vee\; (\Phi \;\wedge\; \forall\bigcirc\forall(\Phi \, \mathsf{U} \, \Psi)) \\
\forall\Diamond\Phi &\equiv \Phi \;\vee\; \forall\bigcirc\forall\Diamond\Phi \\
\forall\Box\Phi &\equiv \Phi \;\wedge\; \forall\bigcirc\forall\Box\Phi
\end{aligned}$$

$$\begin{aligned}
\exists(\Phi \, \mathsf{U} \, \Psi) &\equiv \Psi \;\vee\; (\Phi \;\wedge\; \exists\bigcirc\exists(\Phi \, \mathsf{U} \, \Psi)) \\
\exists\Diamond\Phi &\equiv \Phi \;\vee\; \exists\bigcirc\exists\Diamond\Phi \\
\exists\Box\Phi &\equiv \Phi \;\wedge\; \exists\bigcirc\exists\Box\Phi
\end{aligned}$$

Distributive laws (1)

Recall in LTL:

$$\begin{aligned}
\Box(\varphi \;\wedge\; \psi) &\equiv \Box\varphi \;\wedge\; \Box\psi \\
\Diamond(\varphi \;\vee\; \psi) &\equiv \Diamond\varphi \;\vee\; \Diamond\psi
\end{aligned}$$

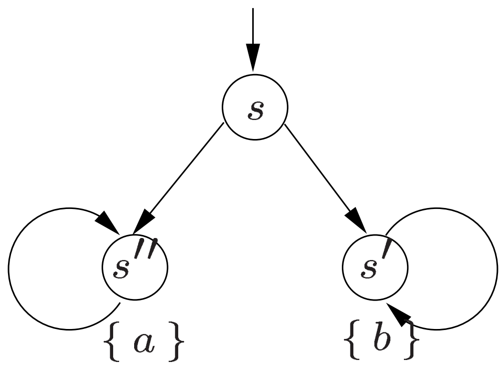
In CTL:

$$\forall\Box(\Phi \wedge \Psi) \;\equiv\; \forall\Box\Phi \;\wedge\; \forall\Box\Psi$$

$$\exists\Diamond(\Phi \vee \Psi) \;\equiv\; \exists\Diamond\Phi \;\vee\; \exists\Diamond\Psi$$

note that $\exists\Box(\Phi \;\wedge\; \Psi) \not\equiv \exists\Box\Phi \;\wedge\; \exists\Box\Psi$ and
$\forall\Diamond(\Phi \;\vee\; \Psi) \not\equiv \forall\Diamond\Phi \;\vee\; \forall\Diamond\Psi$

Distributive laws (2)



$s \models \forall\Diamond(a \vee b)$ since for all $\pi \in$ *Paths*$(s).\ \pi \models \Diamond(a \vee b)$

But: $s\,(s'')^\omega \models \Diamond a$ but $s\,(s'')^\omega \not\models \Diamond b$ Thus: $s \not\models \forall\Diamond b$

A similar reasoning applied to path $s\,(s')^\omega$ yields $s \not\models \forall\Diamond a$

Thus, $s \not\models \forall\Diamond a \vee \forall\Diamond b$

Existential normal form (ENF)

The set of CTL formulas in *existential normal form* (ENF) is given by:

$$\Phi ::= \text{true} \mid a \mid \Phi_1 \wedge \Phi_2 \mid \neg\Phi \mid \exists\bigcirc\Phi \mid \exists(\Phi_1 \,\mathsf{U}\, \Phi_2) \mid \exists\Box\Phi$$

For each CTL formula, there exists an equivalent CTL formula in ENF

$$\forall\bigcirc\Phi \equiv \neg\exists\bigcirc\neg\Phi$$

$$\forall(\Phi \,\mathsf{U}\, \Psi) \equiv \neg\exists(\neg\Psi \,\mathsf{U}\, (\neg\Phi \wedge \neg\Psi)) \wedge \neg\exists\Box\neg\Psi$$

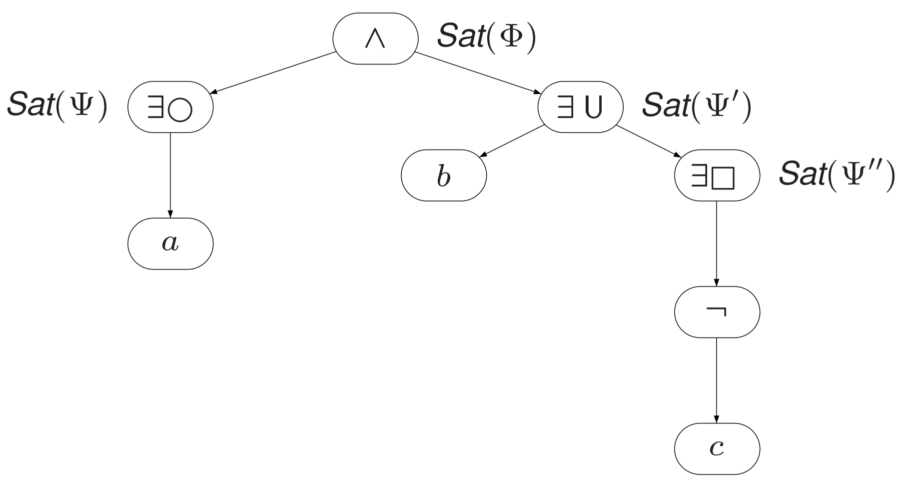
Model checking CTL

- How to check whether state graph *S* satisfies CTL formula $\widehat{\Phi}$?
  - convert the formula $\widehat{\Phi}$ into the equivalent Φ in ENF
  - compute *recursively* the set $Sat(\Phi) = \{ q \in S \mid q \models \Phi \}$
  - $S \models \Phi$ if and only if each initial state of *S* belongs to $Sat(\Phi)$
- Recursive bottom-up computation of $Sat(\Phi)$:
  - consider the parse-tree of Φ
  - start to compute $Sat(a_i)$, for all leafs in the tree
  - then go one level up in the tree and determine $Sat(\cdot)$ for these nodes

$$\text{e.g.,:} \quad Sat(\underbrace{\Psi_1 \wedge \Psi_2}_{\text{node at level } i}) = Sat(\underbrace{\Psi_1}_{\substack{\text{node at}\\ \text{level } i-1}}) \cap Sat(\underbrace{\Psi_2}_{\substack{\text{node at}\\ \text{level } i-1}})$$

  - then go one level up and determine $Sat(\cdot)$ of these nodes
  - and so on....... until the root is treated, i.e., $Sat(\Phi)$ is computed

Example



$$\Phi = \underbrace{\exists\bigcirc a}_{\Psi} \wedge \underbrace{\exists(b \,\mathsf{U}\, \underbrace{\exists\Box \neg c}_{\Psi''})}_{\Psi'} .$$

Basic algorithm

Input: finite state graph *S* and CTL formula Φ (both over *AP*)
Output: $S \models \Phi$

(* compute the sets *Sat*$(\Phi) = \{ q \in Q \mid q \models \Phi \}$ *)

```
for all i ⩽ | Φ | do
   for all Ψ ∈ Sub(Φ) with | Ψ | = i do
      compute Sat(Ψ) from Sat(Ψ')          (* for maximal proper Ψ' ∈ Sub(Ψ) *)
   od
od
return Q₀ ⊆ Sat(Φ)
```


Characterization of *Sat* (1)

For all CTL formulas Φ, Ψ over *AP* it holds:

$$\begin{aligned}
Sat(\text{true}) &= Q \\
Sat(a) &= \{ q \in Q \mid a \in L(q) \}, \text{ for any } a \in AP \\
Sat(\Phi \wedge \Psi) &= Sat(\Phi) \cap Sat(\Psi) \\
Sat(\neg\Phi) &= Q \setminus Sat(\Phi) \\
Sat(\exists \bigcirc \Phi) &= \{ q \in Q \mid Successors(q) \cap Sat(\Phi) \neq \varnothing \}
\end{aligned}$$

where $S = (Q, Q_0, E, L)$ is a finite state graph without terminal states

Characterization of *Sat* (2)

- $\mathit{Sat}(\exists(\Phi \,\mathsf{U}\, \Psi))$ is the smallest subset T of Q, such that:

  (1) $\mathit{Sat}(\Psi) \subseteq T$ and (2) $(q \in \mathit{Sat}(\Phi)$ and $\mathit{Successors}(q) \cap T \neq \varnothing) \;\Rightarrow\; q \in T$

- $\mathit{Sat}(\exists\Box\,\Phi)$ is the largest subset T of Q, such that:

  (3) $T \subseteq \mathit{Sat}(\Phi)$ and (4) $q \in T$ implies $\mathit{Successors}(q) \cap T \neq \varnothing$

where $S = (Q, Q_0, E, L)$ is a state graph without terminal states

Computing $\mathit{Sat}(\exists(\Phi \,\mathsf{U}\, \Psi))$ (1)

- $\mathit{Sat}(\exists(\Phi \,\mathsf{U}\, \Psi))$ is the smallest set $T \subseteq Q$ such that:

  (1) $\mathit{Sat}(\Psi) \subseteq T$ and (2) $(q \in \mathit{Sat}(\Phi)$ and $\mathit{Successors}(q) \cap T \neq \varnothing) \;\Rightarrow\; q \in T$

- This suggests to compute $\mathit{Sat}(\exists(\Phi \,\mathsf{U}\, \Psi))$ iteratively:

  $T_0 = \mathit{Sat}(\Psi)$ and $T_{i+1} = T_i \cup \{\, q \in \mathit{Sat}(\Phi) \mid \mathit{Successors}(q) \cap T_i \neq \varnothing \,\}$

- T_i = states that can reach a Ψ-state in at most i steps via a Φ-path

- By induction on j it follows:

$$T_0 \subseteq T_1 \subseteq \ldots \subseteq T_j \subseteq T_{j+1} \subseteq \;\ldots\; \subseteq \mathit{Sat}(\exists(\Phi \,\mathsf{U}\, \Psi))$$

Computing $Sat(\exists(\Phi \, \mathsf{U} \, \Psi))$ (2)

- S is finite, so for some $j \geqslant 0$ we have: $T_j = T_{j+1} = T_{j+2} = \ldots$
- Therefore: $T_j = T_j \cup \{ q \in Sat(\Phi) \mid Successors(q) \cap T_j \neq \varnothing \}$
- Hence: $\{ q \in Sat(\Phi) \mid Successors(q) \cap T_j \neq \varnothing \} \subseteq T_j$
  - hence, T_j satisfies (2), i.e., $(q \in Sat(\Phi)$ and $Successors(q) \cap T_j \neq \varnothing) \Rightarrow q \in T_j$
  - further, $Sat(\Psi) = T_0 \subseteq T_j$ so, T_j satisfies (1), i.e. $Sat(\Psi) \subseteq T_j$
- As $Sat(\exists(\Phi \, \mathsf{U} \, \Psi))$ is the *smallest* set satisfying (1) and (2):
  - $Sat(\exists(\Phi \, \mathsf{U} \, \Psi)) \subseteq T_j$ and thus $Sat(\exists(\Phi \, \mathsf{U} \, \Psi)) = T_j$
- Hence: $T_0 \subsetneqq T_1 \subsetneqq T_2 \subsetneqq \ldots \subsetneqq T_j = T_{j+1} = \ldots = Sat(\exists(\Phi \, \mathsf{U} \, \Psi))$

Computing $Sat(\exists(\Phi \, \mathsf{U} \, \Psi))$ (3)

Input: finite state graph S with state-set Q and CTL-formula $\exists(\Phi \, \mathsf{U} \, \Psi)$
Output: $Sat(\exists(\Phi \, \mathsf{U} \, \Psi)) = \{ q \in Q \mid q \models \exists(\Phi \, \mathsf{U} \, \Psi) \}$

```
V := Sat(Ψ);       (* V administers states q with q ⊨ ∃(Φ U Ψ) *)
T := V;            (* T contains the already visited states q with q ⊨ ∃(Φ U Ψ) *)
while V ≠ ∅ do
   let q' ∈ V;
   V := V \ { q' };
   for all q ∈ Pre(q') do
      if q ∈ Sat(Φ) \ T then V := V ∪ { q }; T := T ∪ { q }; endif
   od
od
return T
```