FROM REAL-TIME LOGIC TO TIMED AUTOMATA

DEJAN NIČKOVIĆ – AIT AUSTRIAN INSTITUTE OF TECHNOLOGY

Joint work with Thomas Ferrère, Oded Maler and Amir Pnueli

MOVEP'20

MOTIVATION

- CPS often have **hard real-time** constraints
- Real-time specification language interpreted over dense time
  - Metric Interval Temporal Logic (MITL)
  - *[AFH96] R. Alur, T. Feder, T. Henzinger: The Benefits of Relaxing Punctuality (1996)*
- Need for an RV approach to evaluate such specs
  - Timed automata (TA) as real-time observers

- How to translate MITL to TA?
  - **Monolithic** tableau construction in [AFH96]

Our contribution

- Novel **compositional** translation
  - **Temporal testers** instead of acceptor automata
  - Facilitates adding extensions
  - Handles past operators for free

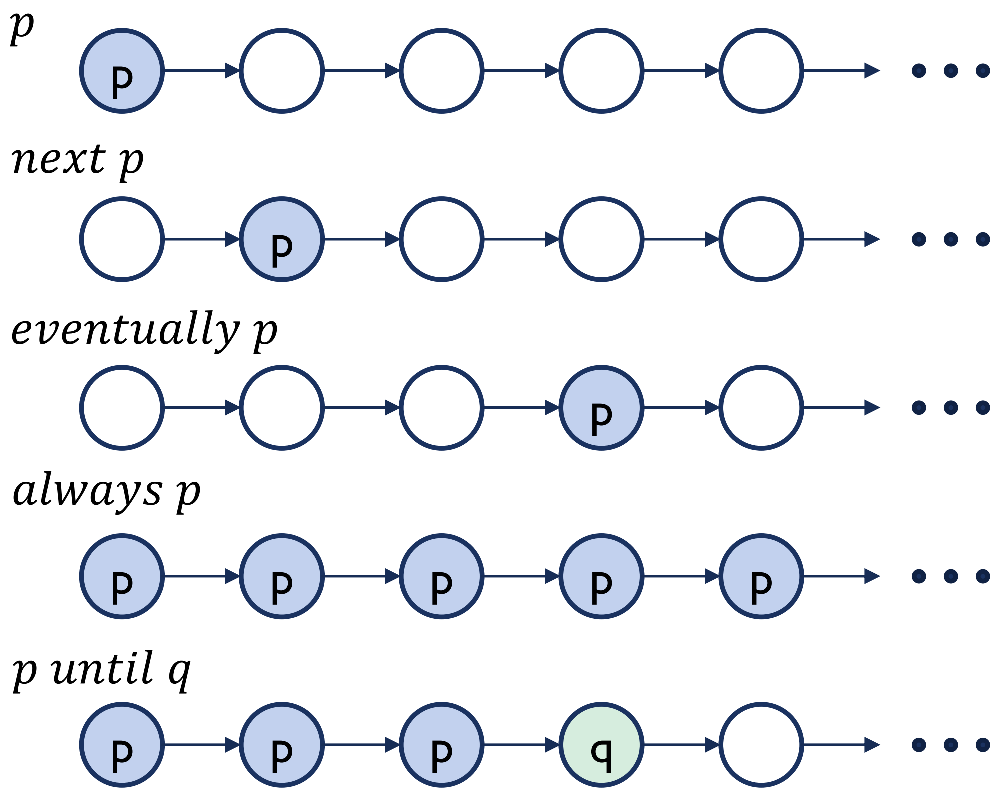
LINEAR TEMPORAL LOGIC (LTL)

Syntax

$$S \coloneqq p \mid not\ S \mid S_1\ or\ S_2 \mid next\ S \mid S_1\ until\ S_2$$

$true$	$\equiv$	$S\ or\ not\ S$
$false$	$\equiv$	$not\ true$
$S_1\ and\ S_2$	$\equiv$	$\text{not}\ (not\ S_1\ or\ not\ S_2)$
$S_1\ implies\ S_2$	$\equiv$	$not\ S_1\ or\ S_2$
$eventually\ S$	$\equiv$	$true\ until\ S$
$always\ S$	$\equiv$	$not\ eventually\ not\ S$

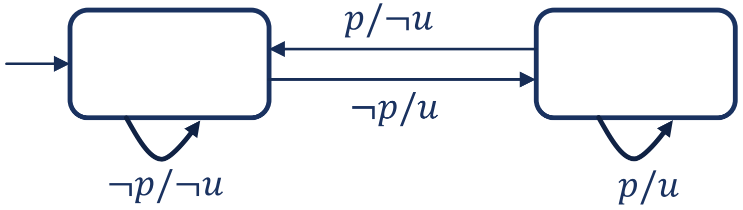
Semantics



WHAT ARE TEMPORAL TESTERS?

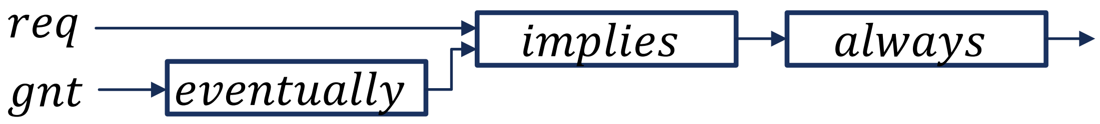
Temporal testers

- **Non-deterministic** sequential transducers
  - Inputs and outputs



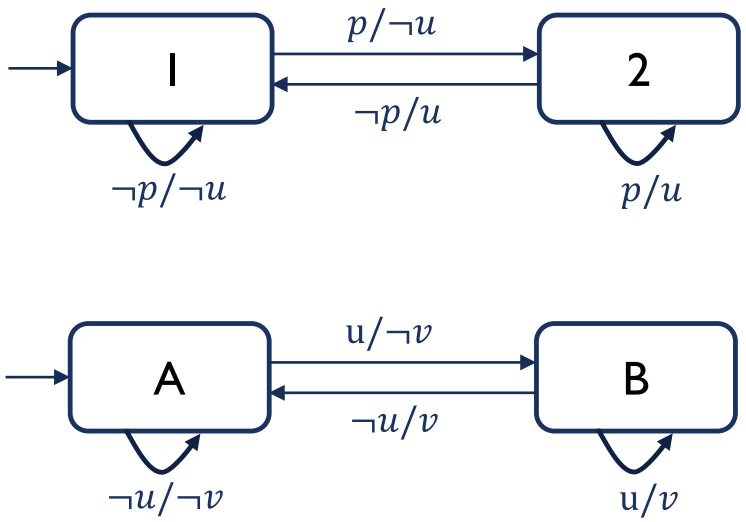
- Realize a temporal logic function
  - Tester T_S for specification S
  - $T_S(w,t) = 1 \leftrightarrow (w,t) \vDash S$

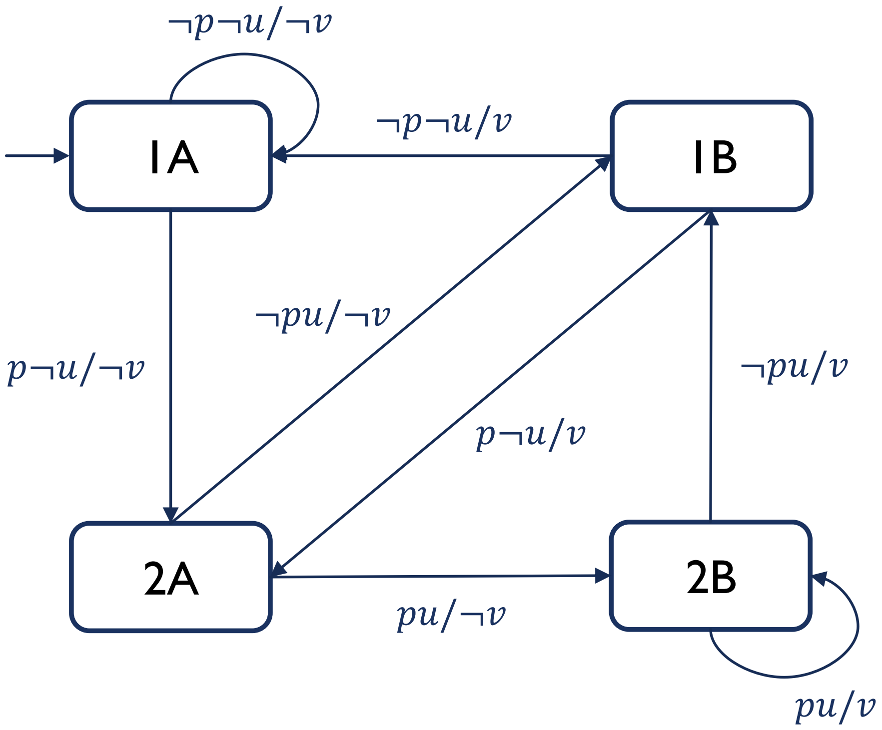
- Modular construction
  - Temporal testers defined for basic temporal operators
  - Arbitrary formula = composition
  - Example: $always(req\ implies\ eventually\ gnt)$



COMPOSITION OF TEMPORAL TESTERS

- Composition of Transducers – Synchronization on I/O





TEMPORAL TESTERS VS. ACCEPTORS

- Acceptor A_S
  - Accepts trace w iff $(\boldsymbol{w}, \mathbf{0}) \vDash \boldsymbol{S}$
- Temporal Tester T_S
  - Checks if $(\boldsymbol{w}, \boldsymbol{t}) \vDash \boldsymbol{S}$ **for all** $\boldsymbol{t} \geq \mathbf{0}$
  - Acceptor for every suffix of w

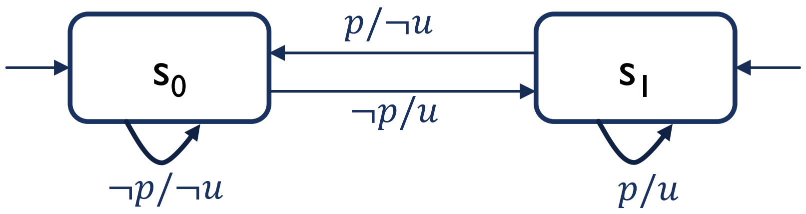
- Given acceptors A_{S_1} and A_{S_2} for S_1 and S_2
  - No simple recipy to compose them to get acceptor for $S_1 until\ S_2$
- Given temporal testers T_{S_1} and T_{S_2} for S_1 and S_2
  - Composition yields tester for $S_1 until\ S_2$

time	0	1	2	3	4
S_1	1	1	1	1	0
S_2	0	0	1	0	0
A_{S_1}	1	?	?	?	?
A_{S_2}	0	?	?	?	?
$A_{S_1 until S_2}$	?	?	?	?	?
T_{S_1}	1	1	1	1	0
T_{S_2}	0	0	1	0	0
$T_{S_1 until S_2}$	1	1	1	0	0

TEMPORAL TESTER FOR NEXT

- Non-causal shift register

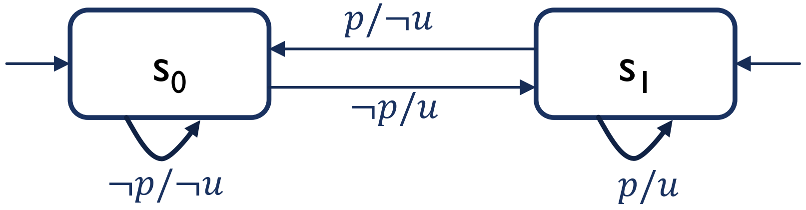
$$u = next\ p$$



TEMPORAL TESTER FOR NEXT

- Non-causal shift register

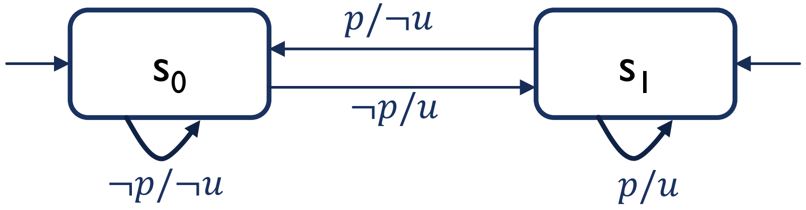
$$u = next\ p$$

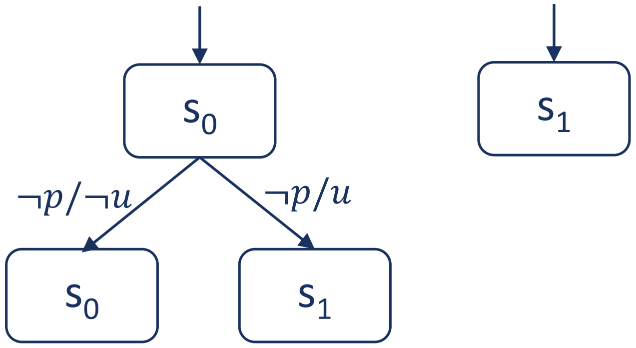


TEMPORAL TESTER FOR NEXT

- Non-causal shift register

$$u = next\ p$$

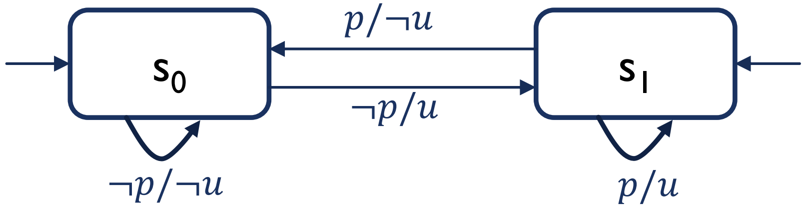


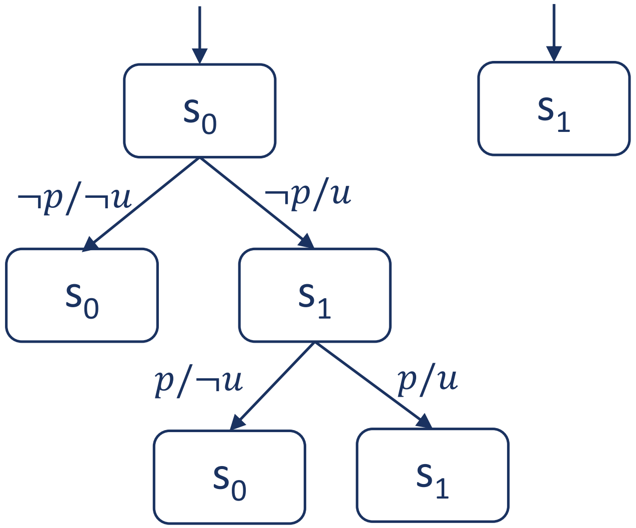


TEMPORAL TESTER FOR NEXT

- Non-causal shift register

$$u = next\ p$$

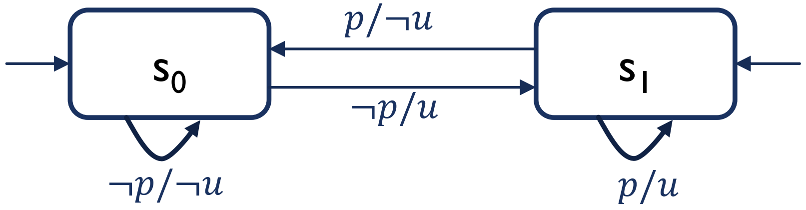


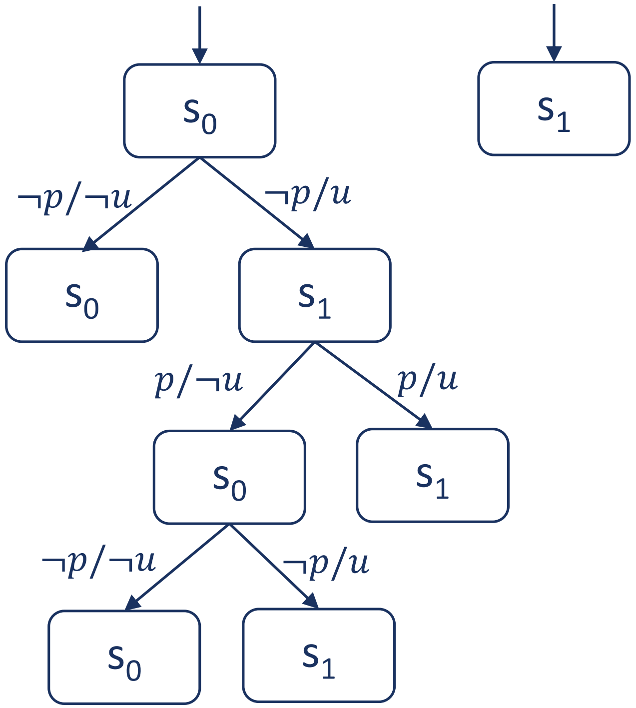


TEMPORAL TESTER FOR NEXT

- Non-causal shift register

$$u = next\ p$$

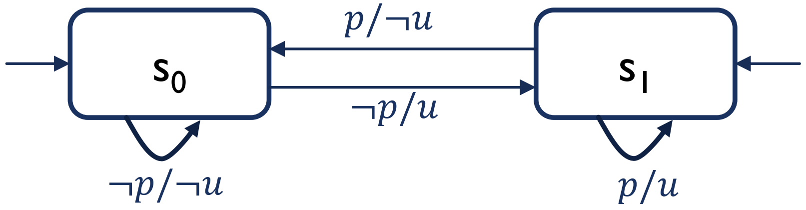


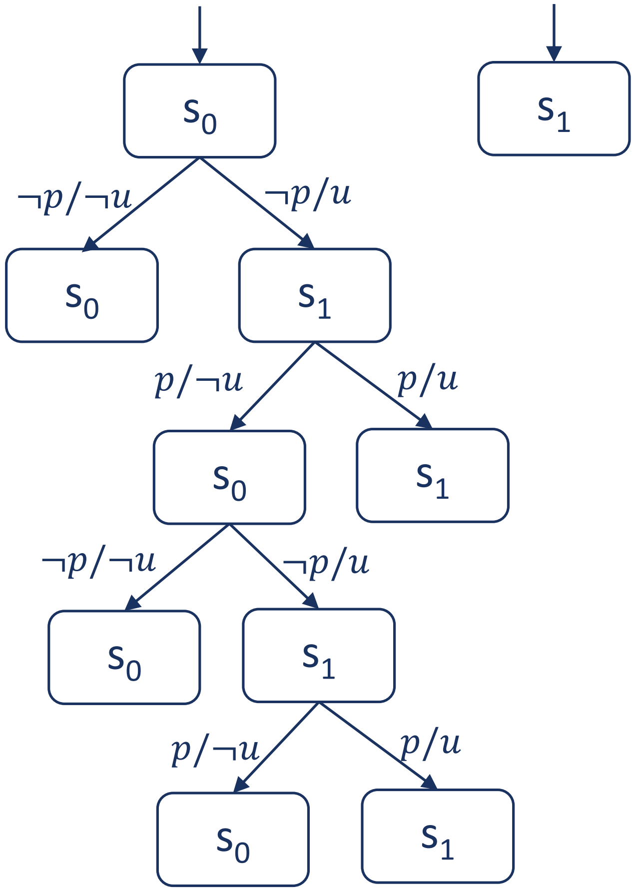


TEMPORAL TESTER FOR NEXT

- Non-causal shift register

$$u = next\ p$$





METRIC INTERVAL TEMPORAL LOGIC (MITL)

- Temporal logic over **continuous-time** signals

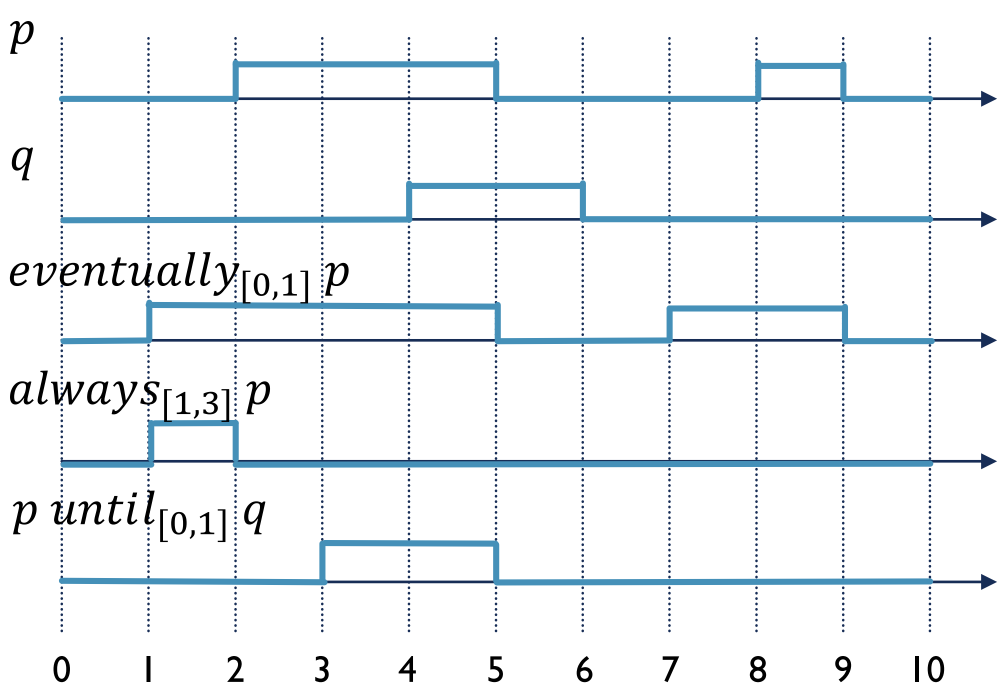
- **Syntax**

$$S \coloneqq p \mid not\ S \mid S_1\ or\ S_2 \mid S_1\ until_I\ S_2$$

- I is a **non-punctual** interval
- Other Boolean and temporal operators derived as expected
- We can also consider MITL with past operators

- **Satisfiability** of MITL is **EXPSPACE-complete**.

- **Semantics**



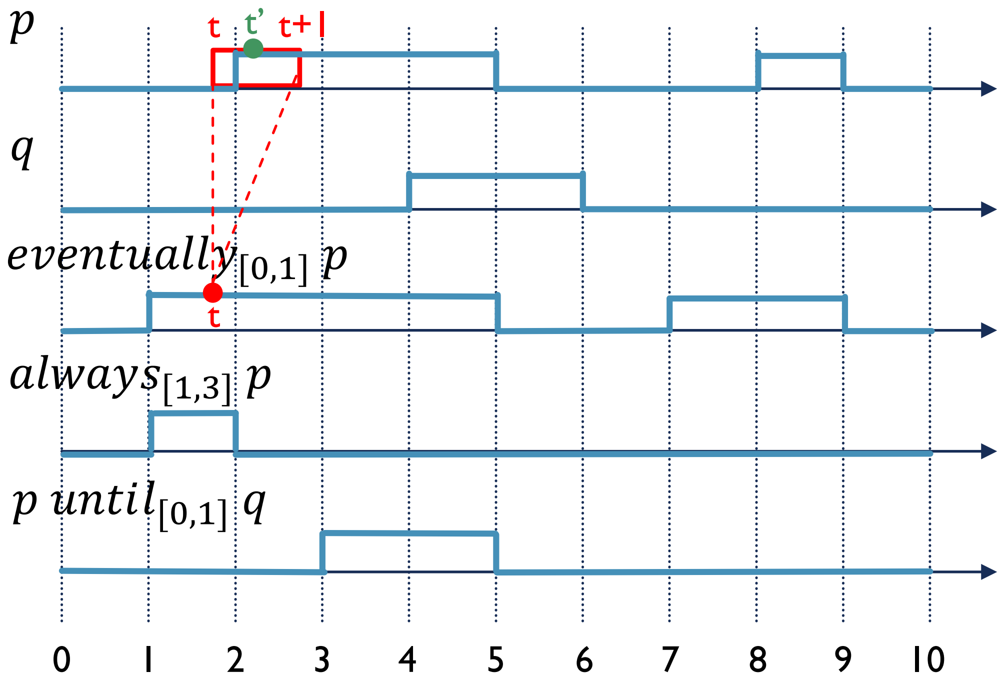
METRIC INTERVAL TEMPORAL LOGIC (MITL)

- Temporal logic over **continuous-time** signals
- **Syntax**

$$S \;\coloneqq p \mid not\ S \mid S_1\ or\ S_2 \mid S_1\ until_I\ S_2$$

- I is a **non-punctual** interval
- Other Boolean and temporal operators derived as expected
- We can also consider MITL with past operators
- **Satisfiability** of MITL is **EXPSPACE-complete**.

- **Semantics**



METRIC INTERVAL TEMPORAL LOGIC (MITL)

- Temporal logic over **continuous-time** signals

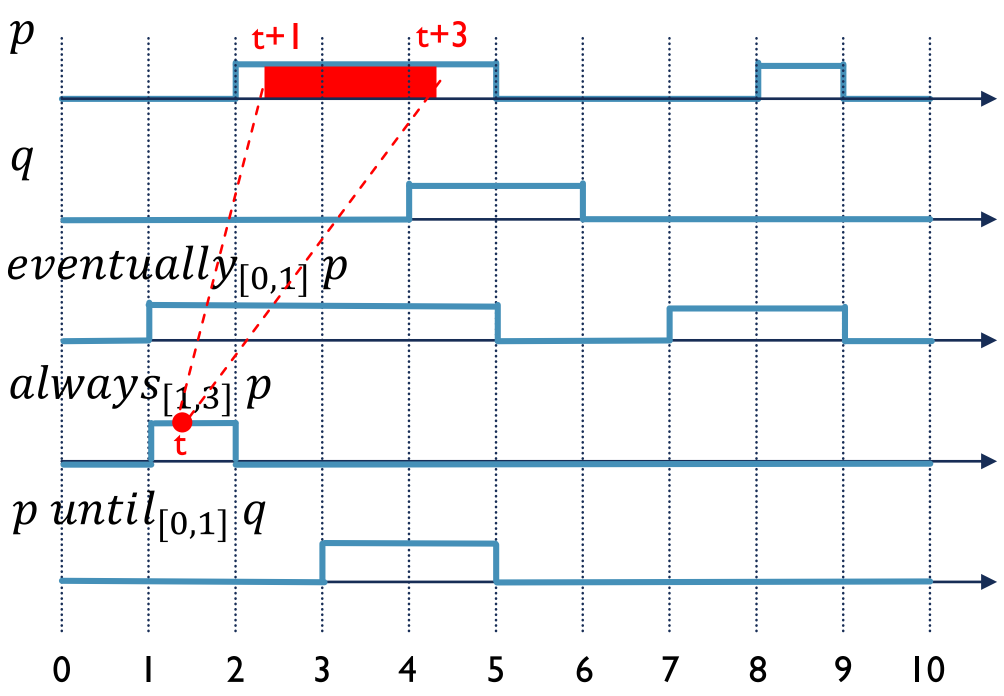
- **Syntax**

$$S \;\coloneqq p \mid not\ S \mid S_1\ or\ S_2 \mid S_1\ until_I\ S_2$$

- I is a **non-punctual** interval
- Other Boolean and temporal operators derived as expected
- We can also consider MITL with past operators

- **Satisfiability** of MITL is **EXPSPACE-complete**.

- **Semantics**



METRIC INTERVAL TEMPORAL LOGIC (MITL)

- Temporal logic over **continuous-time** signals

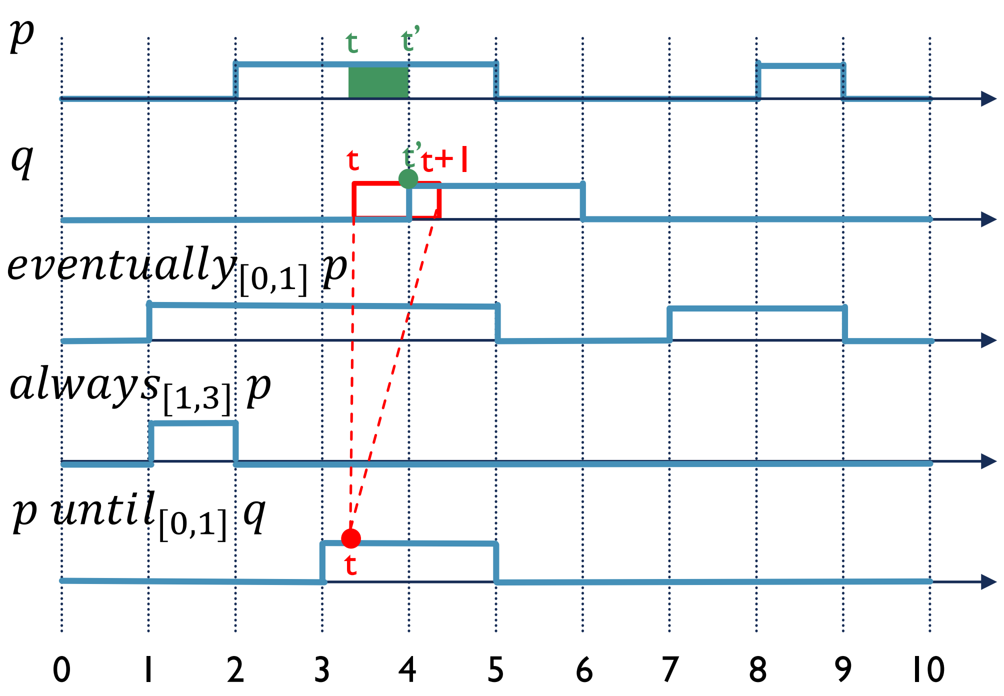
- **Syntax**

$$S \coloneqq p \mid not\ S \mid S_1\ or\ S_2 \mid S_1\ until_I\ S_2$$

- I is a **non-punctual** interval
- Other Boolean and temporal operators derived as expected
- We can also consider MITL with past operators

- **Satisfiability** of MITL is **EXPSPACE-complete**.

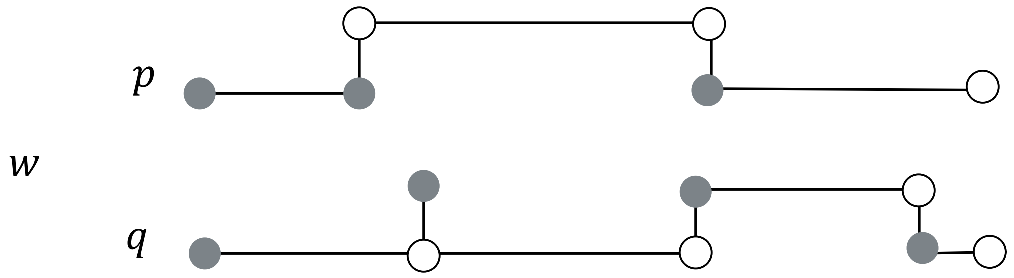
- **Semantics**



SOME ADDITIONAL NOTES ON MITL

Signals

- Multi-dimensional Boolean signals
  - $w : R_+ \to B^n$
- Alternating sequence of singular points and open intervals

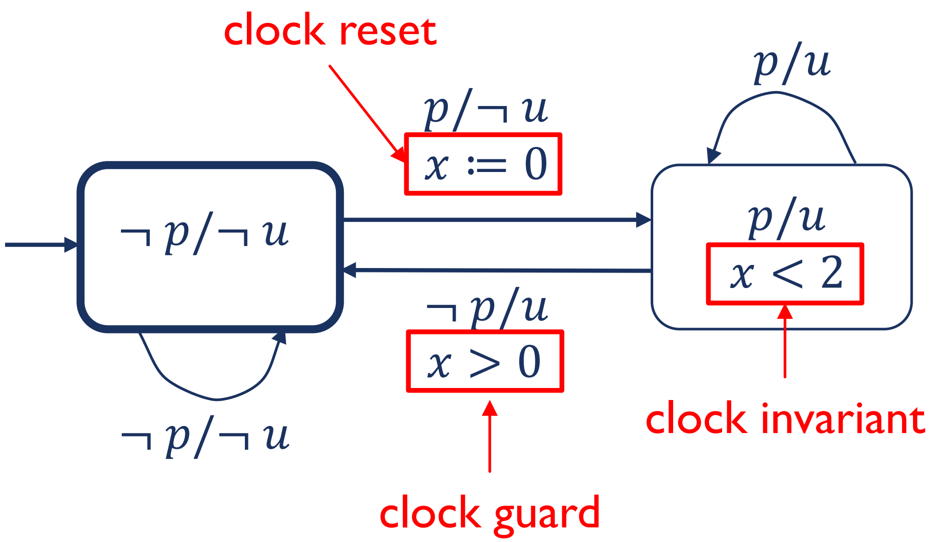


Semantics

- Strict definition of until
- $(w,t) \vDash S_1 until_I S_2 \leftrightarrow \exists t' \in t \oplus I \ s.t. (w,t') \vDash S_2 \ and \ \forall t'' \in (t,t'), (w,t'') \vDash S_1$
- $S_1 \ until \ S_2 = S_1 \ until_{(0,\infty)} \ S_2$
- $next \ S = S \ until \ S$

TIMED TEMPORAL TESTERS

- TA with inputs and outputs
- Auxiliary **clocks**
  - Location **invariants**
  - Transition **guards** and **resets**
- Labels on both transitions and locations
- Generalized Büchi conditions on locations

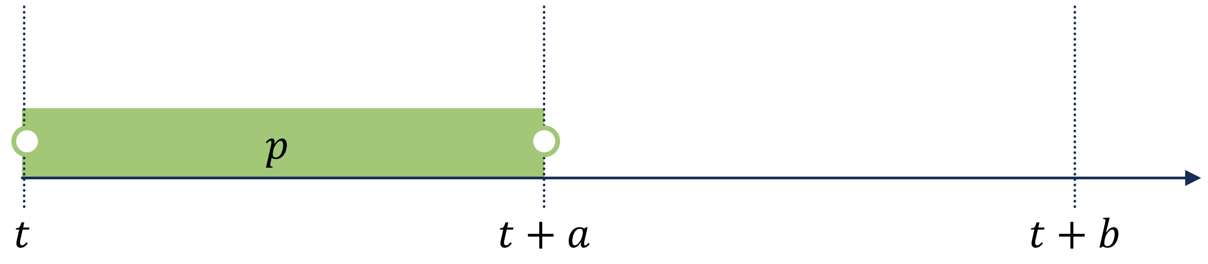


FROM MITL TO AUTOMATA – FORMULA SIMPLIFICATION

- We only need a temporal tester for $until_I$!
- Building temporal testers for $until_I$ is hard
  - Simplify specifications

- Eliminate $until_I$
  - $p\ until_{(a,b)} q = p\ until_{(a,\infty)} q\ and\ eventually_{(a,b)} q$
  - $p\ until_{(c,\infty)} q = always_{(0,c]}(p\ and\ p\ until\ q)$

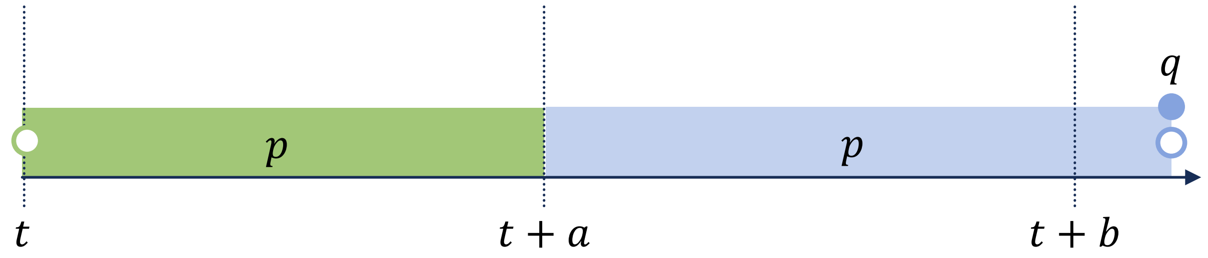
$$p\ until_{(a,b)} q = always_{(0,a)} p \wedge always_{(0,a]}(p\ until\ q) \wedge eventually_{(a,b)} q$$



FROM MITL TO AUTOMATA – FORMULA SIMPLIFICATION

- We only need a temporal tester for $until_I$!
- Building temporal testers for $until_I$ is hard
  - Simplify specifications
- Eliminate $until_I$
  - $p\ until_{(a,b)} q = p\ until_{(a,\infty)} q\ and\ eventually_{(a,b)} q$
  - $p\ until_{(c,\infty)} q = always_{(0,c]}(p\ and\ p\ until\ q)$

$$p\ until_{(a,b)} q = always_{(0,a)} p \wedge always_{(0,a]}(p\ until\ q) \wedge eventually_{(a,b)} q$$

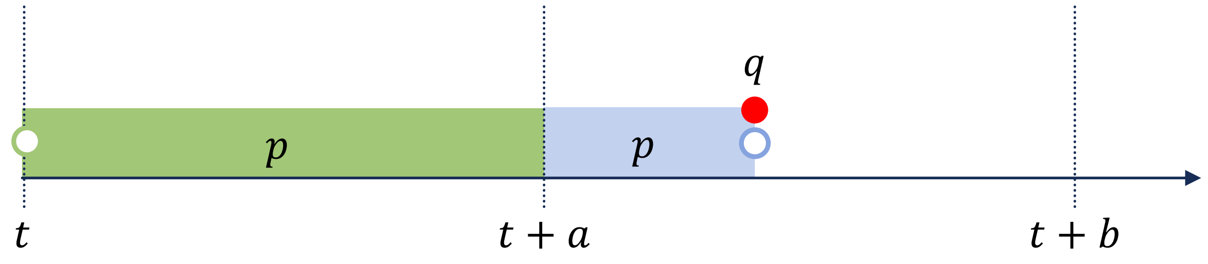


FROM MITL TO AUTOMATA – FORMULA SIMPLIFICATION

- We only need a temporal tester for $until_I$!
- Building temporal testers for $until_I$ is hard
  - Simplify specifications

- Eliminate $until_I$
  - $p\ until_{(a,b)} q = p\ until_{(a,\infty)} q\ and\ eventually_{(a,b)} q$
  - $p\ until_{(c,\infty)} q = always_{(0,c]}(p\ and\ p\ until\ q)$

$$p\ until_{(a,b)} q = always_{(0,a)} p \wedge always_{(0,a]}(p\ until\ q) \wedge eventually_{(a,b)} q$$



FROM MITL TO AUTOMATA – FORMULA SIMPLIFICATION

- We only need a temporal tester for $until_I$!
- Building temporal testers for $until_I$ is hard
  - Simplify specifications

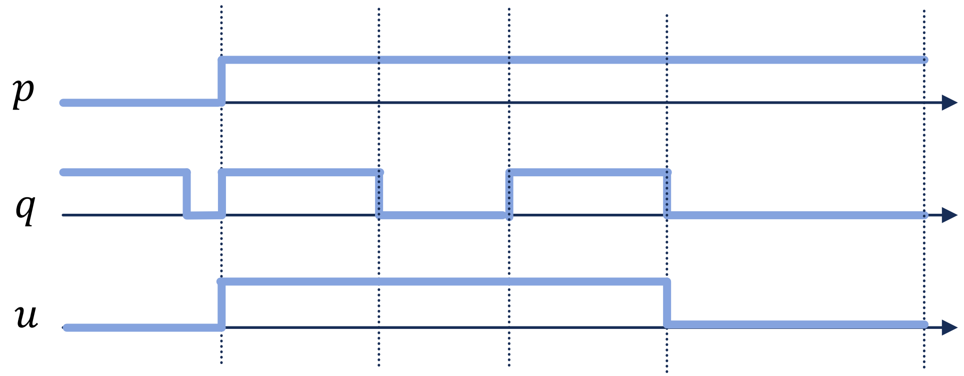
- Eliminate $until_I$
  - $p\ until_{(a,b)} q = p\ until_{(a,\infty)} q\ and\ eventually_{(a,b)} q$
  - $p\ until_{(c,\infty)} q = always_{(0,c]}(p\ and\ p\ until\ q)$
- Eliminate $eventually$ with positive lower bound
  - $eventually_{(a+c,b+c)}\ p =$
    $eventually_{(0,c)}\ always_{(0,c)}\ eventually_{(a,b)}\ p$
  - $eventually_{(0,a]}\ p =$
    $eventually_{(0,a)}\ or\ (next\ always_{(0,a)}\ and\ (not\ p)\ until\ p)$

It is sufficient to build temporal testers for $until$ and $\boldsymbol{eventually_{(0,a)}}$

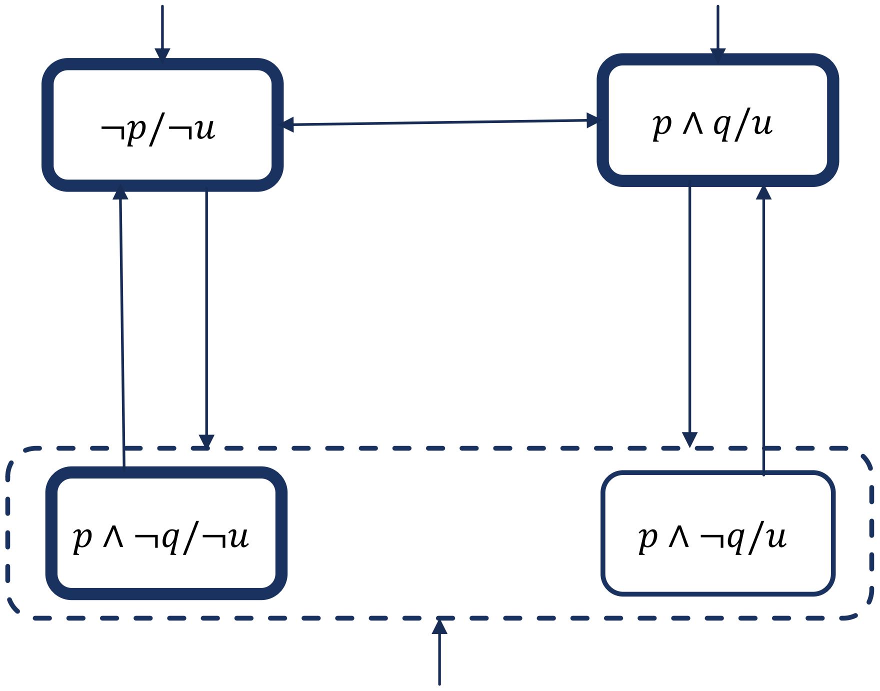
TEMPORAL TESTER FOR UNTIL

Intuition

- Assume
  - Signals with left-closed right-open intervals only
  - Non-strict semantics of until
    - $p\ \underline{until}\ q = q\ or\ (p\ and\ p\ until\ q)$



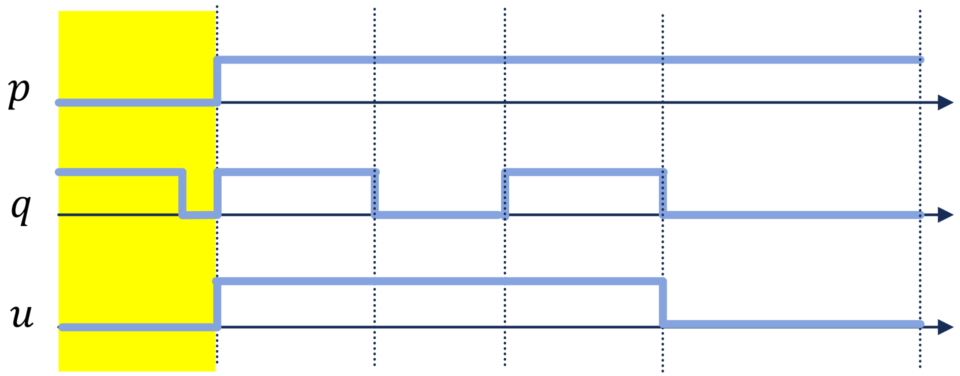
$u = p\ \underline{until}\ q$



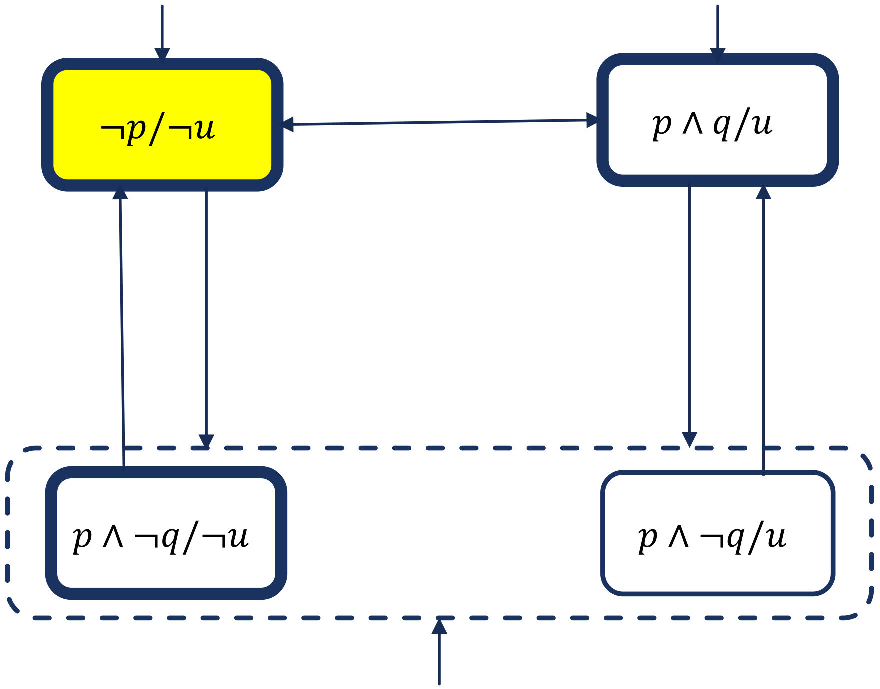
TEMPORAL TESTER FOR UNTIL

Intuition

- Assume
  - Signals with left-closed right-open intervals only
  - Non-strict semantics of until



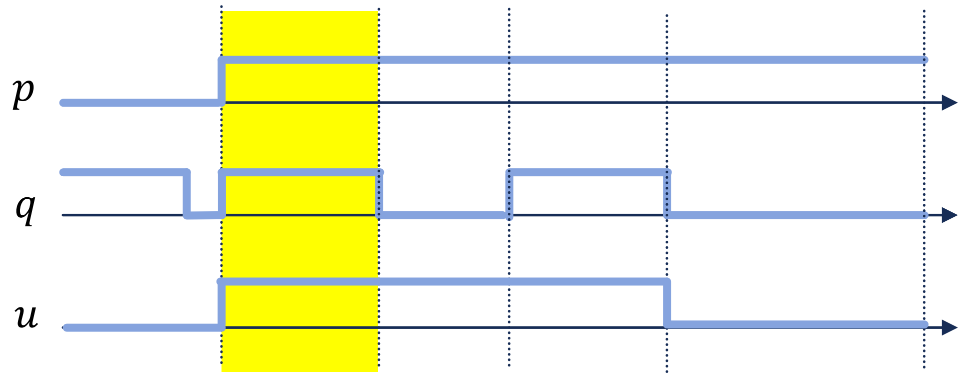
$u = p\ \underline{until}\ q$



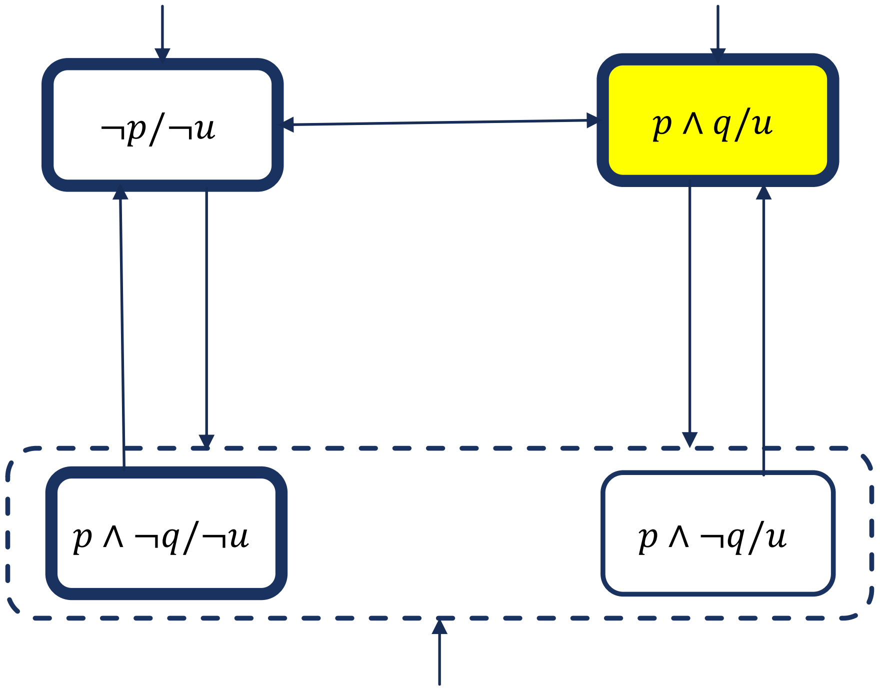
TEMPORAL TESTER FOR UNTIL

Intuition

- Assume
  - Signals with left-closed right-open intervals only
  - Non-strict semantics of until
    - $p\ \underline{until}\ q = (p\ and\ q)\ or\ (p\ and\ p\ until\ q)$



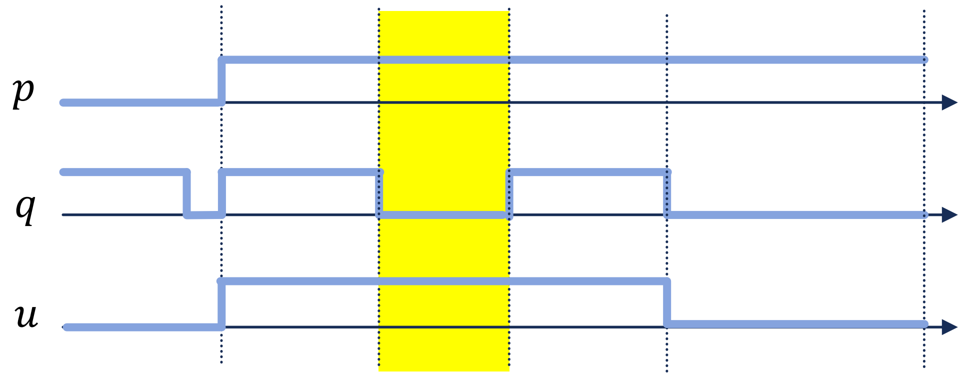
$u = p\ \underline{until}\ q$



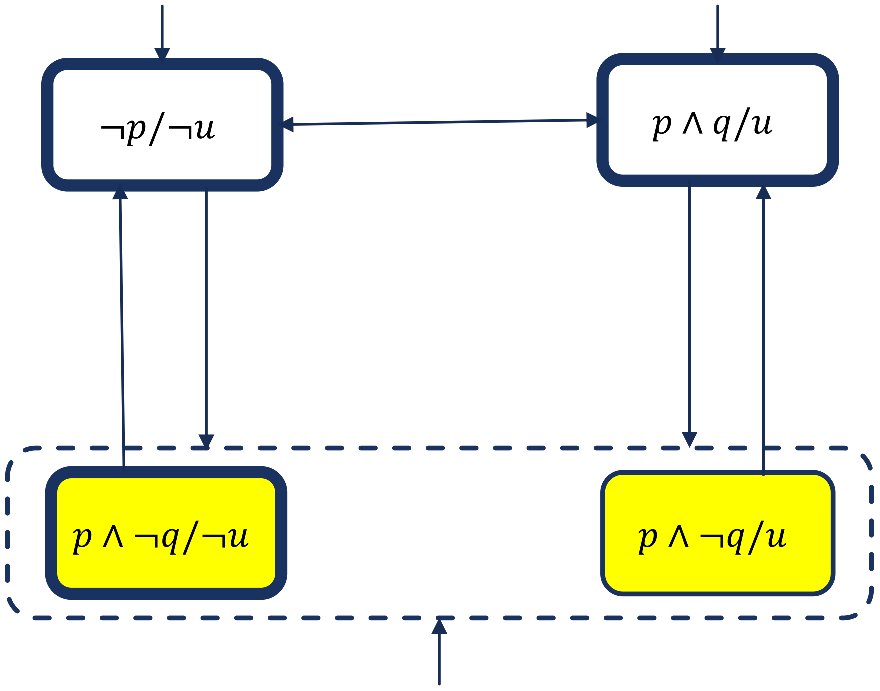
TEMPORAL TESTER FOR UNTIL

Intuition

- Assume
  - Signals with left-closed right-open intervals only
  - Non-strict semantics of until
    - $p \,\underline{until}\, q = q \text{ or } (p \text{ and } p \text{ until } q)$



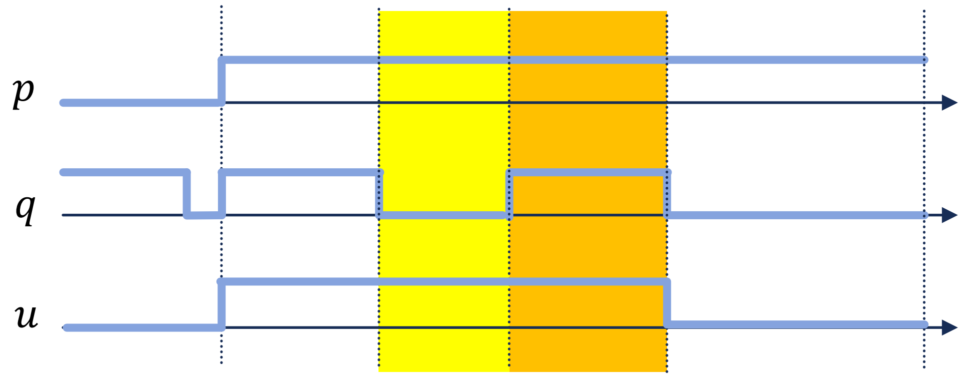
$u = p \,\underline{until}\, q$



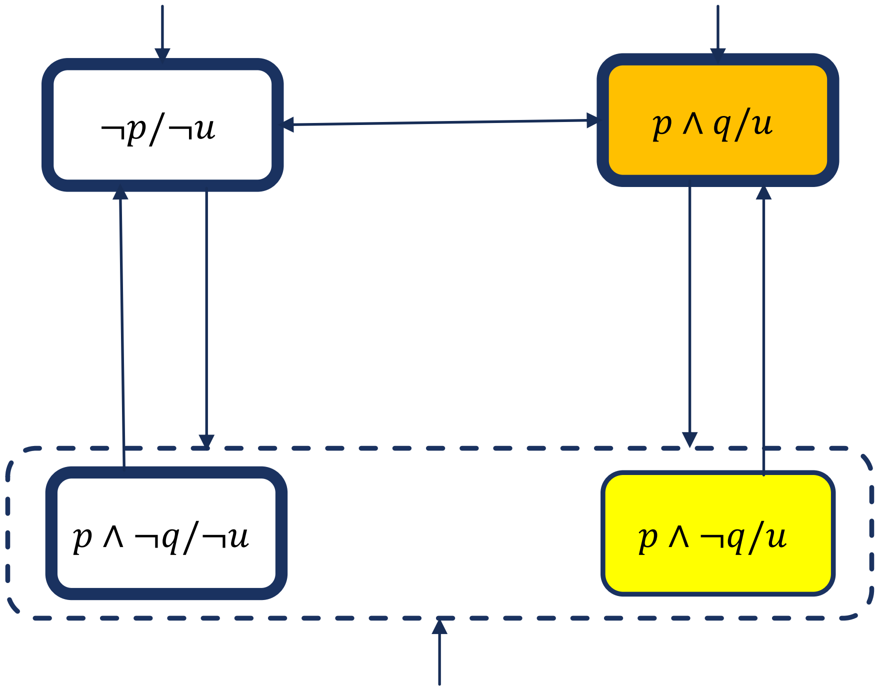
TEMPORAL TESTER FOR UNTIL

Intuition

- Assume
  - Signals with left-closed right-open intervals only
  - Non-strict semantics of until
    - $p\ \underline{until}\ q = q\ or\ (p\ and\ p\ until\ q)$



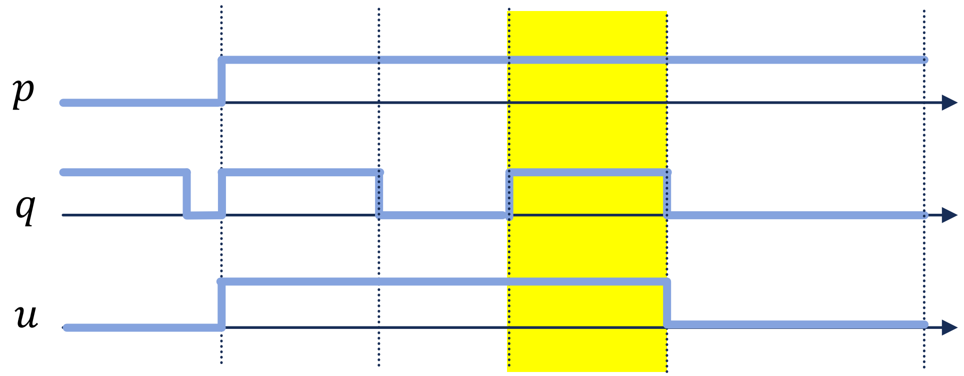
$u = p\ \underline{until}\ q$



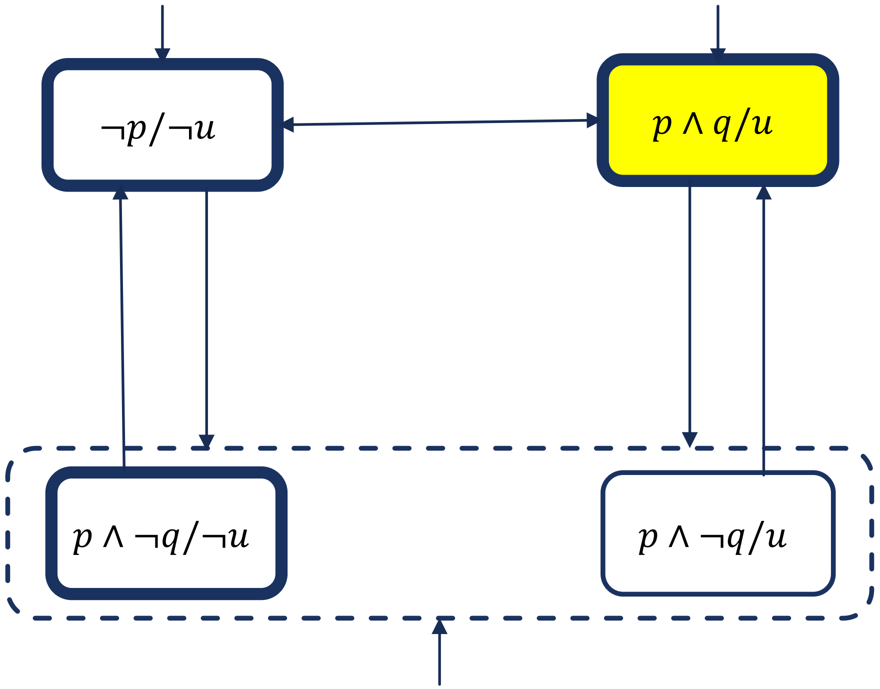
TEMPORAL TESTER FOR UNTIL

Intuition

- Assume
  - Signals with left-closed right-open intervals only
  - Non-strict semantics of until
    - $p\ \underline{until}\ q = q\ or\ (p\ and\ p\ until\ q)$



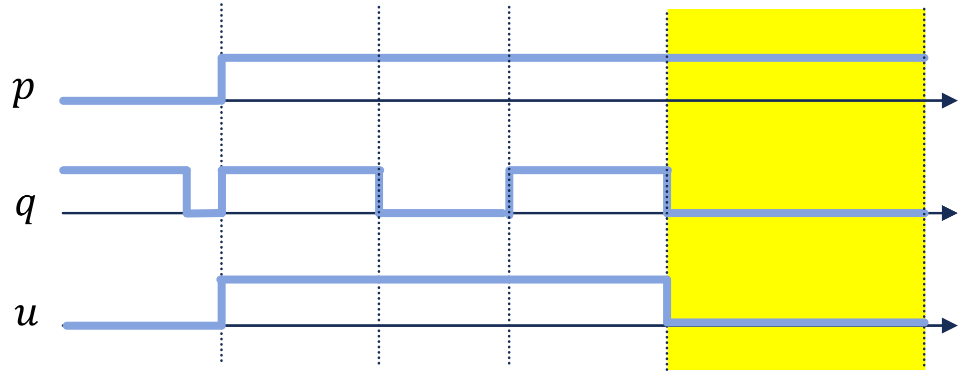
$u = p\ \underline{until}\ q$



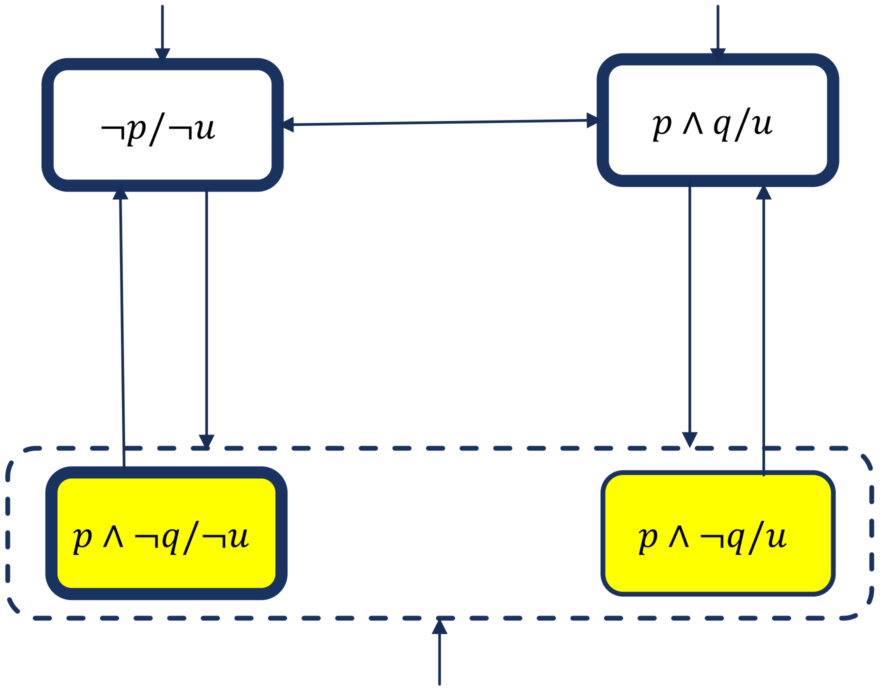
TEMPORAL TESTER FOR UNTIL

Intuition

- Assume
  - Signals with left-closed right-open intervals only
  - Non-strict semantics of until
    - $p\ \underline{until}\ q = q\ or\ (p\ and\ p\ until\ q)$



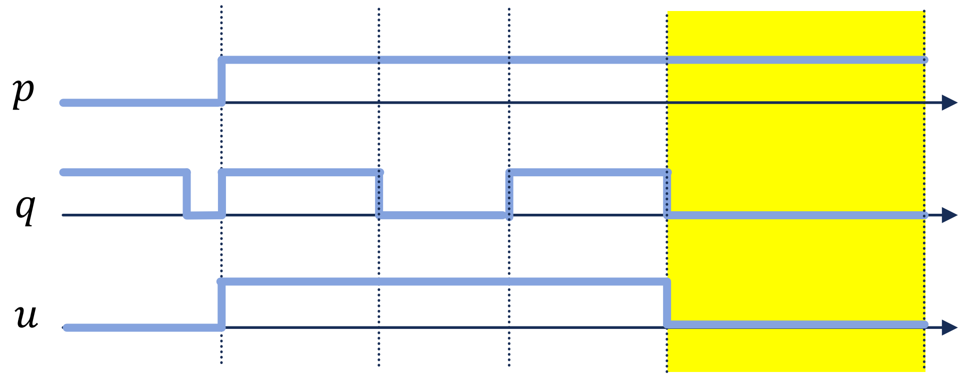
$u = p\ \underline{until}\ q$



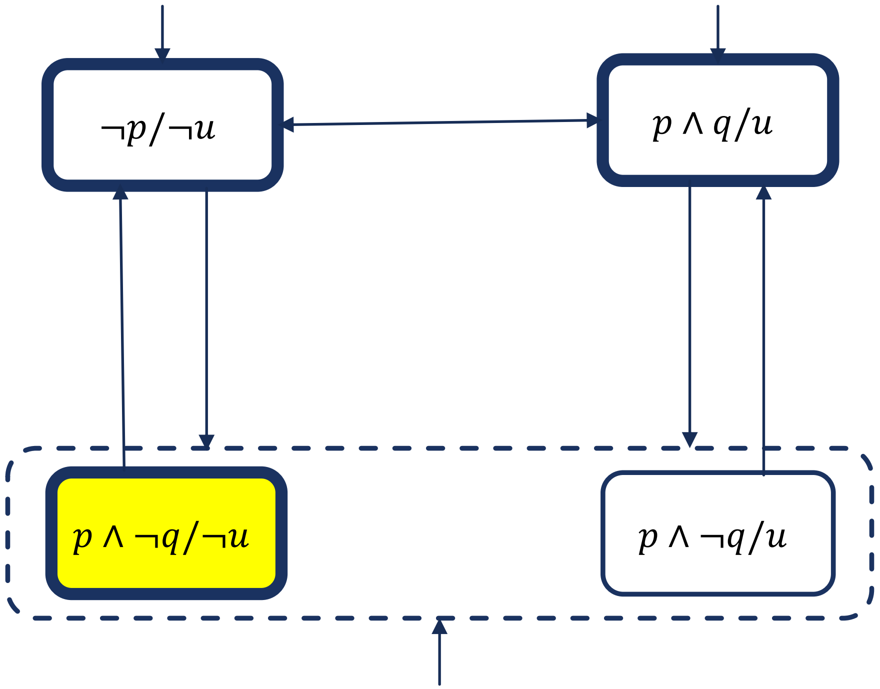
TEMPORAL TESTER FOR UNTIL

Intuition

- Assume
  - Signals with left-closed right-open intervals only
  - Non-strict semantics of until
    - $p\ \underline{until}\ q = q\ or\ (p\ and\ p\ until\ q)$



$$u = p\ \underline{until}\ q$$



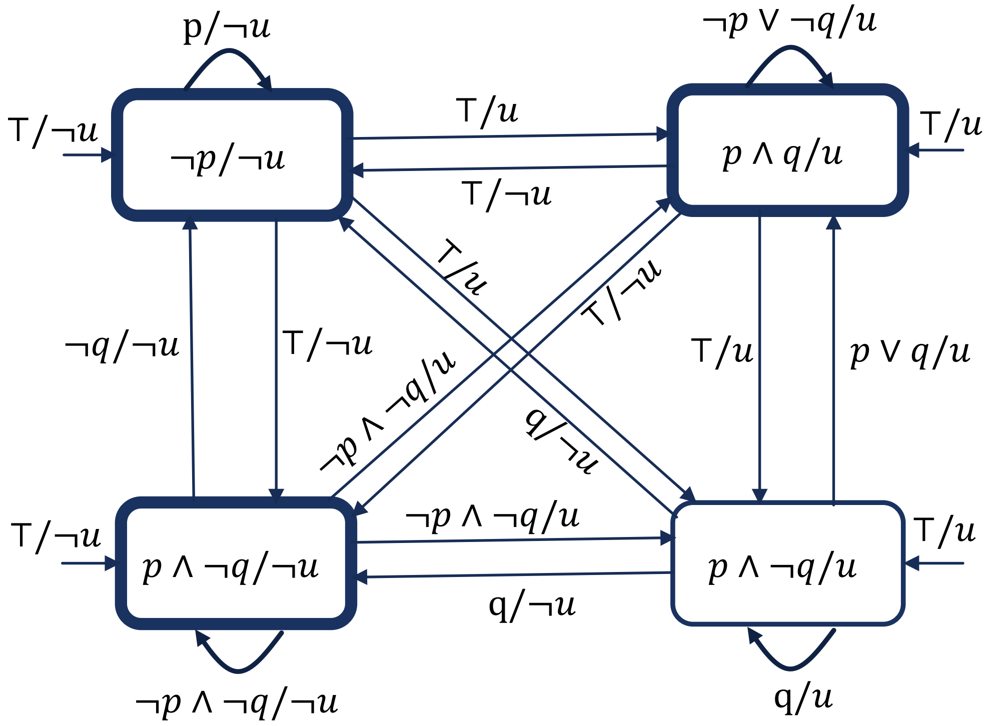
TEMPORAL TESTER FOR UNTIL

Full construction

- $p\ until\ q$ is **right-continuous**
  - $(w,t) \vDash p\ until\ q \rightarrow \exists t' > t, \forall t'' \in (t',t)\ (w,t'') \vDash p\ until\ q$
  - $u = u(t_0) \cdot u(t_0,t_1) \cdot u(t_1) \cdot u(t_1,t_2) \cdots$
  - $u(t_i) = u(t_i,t_{i+1}), \forall i \geq 0$

$w(t_i)$	$w(t_i,t_{i+1})$	$u(t_i,t_{i+1})$
$*$	$p \wedge q$	1
$*$	$\neg p$	0
q	$p \wedge \neg q$	1
$\neg p \wedge \neg q$	$p \wedge \neg q$	0
$p \wedge \neg q$	$p \wedge \neg q$	$u(t_i)$

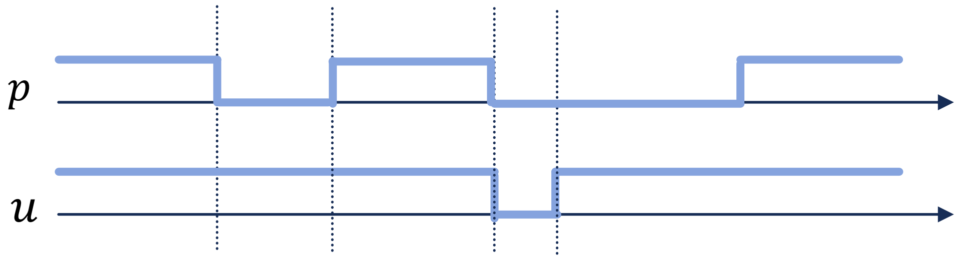
$u = p\ until\ q$



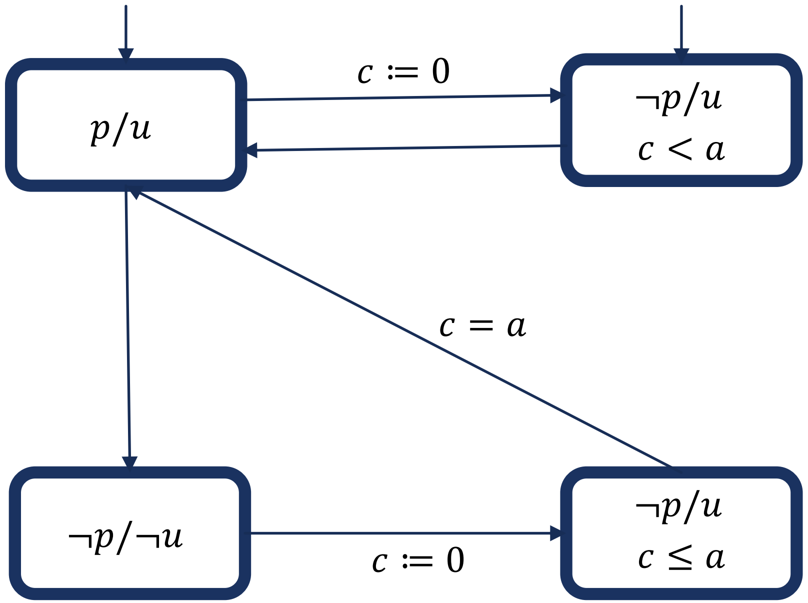
TEMPORAL TESTER FOR EVENTUALLY

Intuition

- Assume
  - Signals with left-closed right-open intervals only
  - Formula of the form $eventually_{[0,a]}\ p$



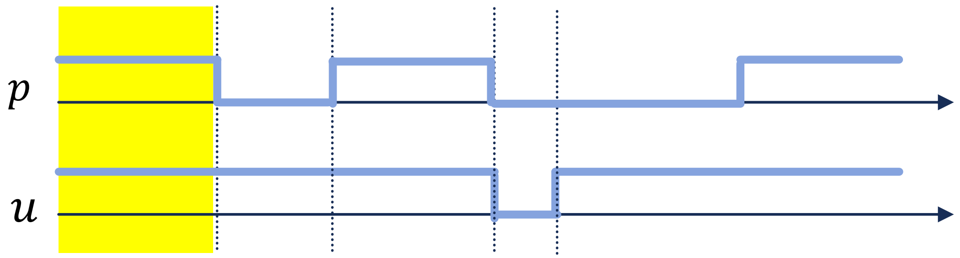
$$u = eventually_{[0,a]}\ p$$



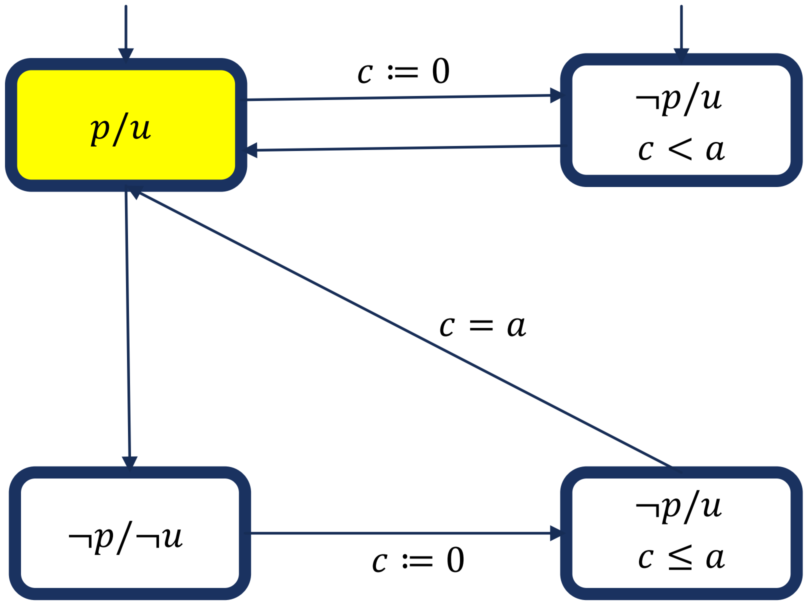
TEMPORAL TESTER FOR EVENTUALLY

Intuition

- Assume
  - Signals with left-closed right-open intervals only
  - Formula of the form $eventually_{[0,a]}\ p$



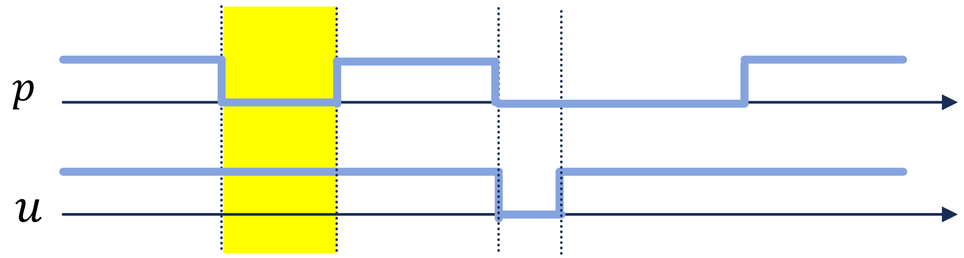
$u = eventually_{[0,a]}\ p$



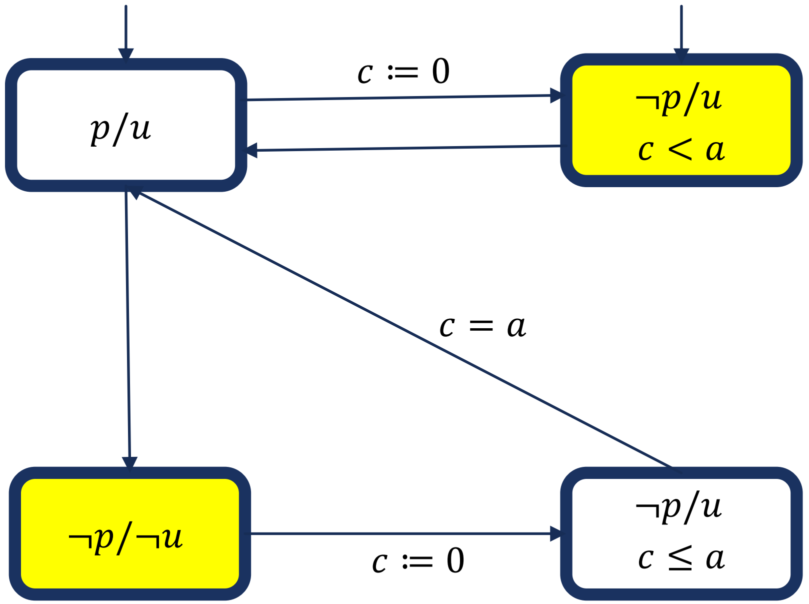
TEMPORAL TESTER FOR EVENTUALLY

Intuition

- Assume
  - Signals with left-closed right-open intervals only
  - Formula of the form $eventually_{[0,a]}\ p$



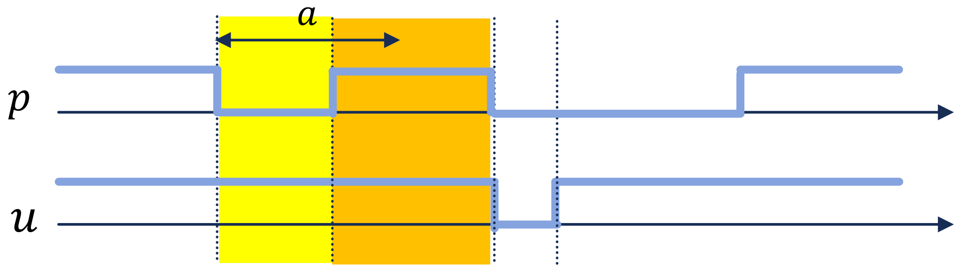
$$u = eventually_{[0,a]}\ p$$



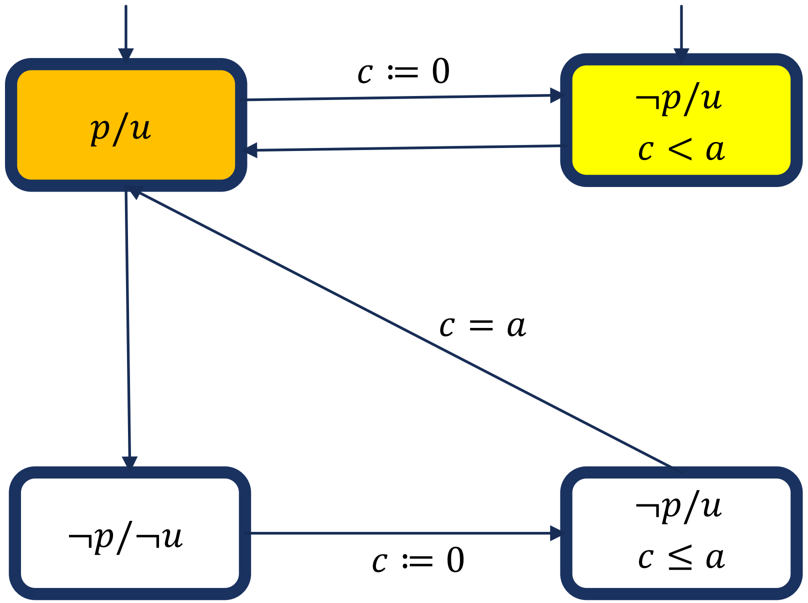
TEMPORAL TESTER FOR EVENTUALLY

Intuition

- Assume
  - Signals with left-closed right-open intervals only
  - Formula of the form $eventually_{[0,a]}\ p$



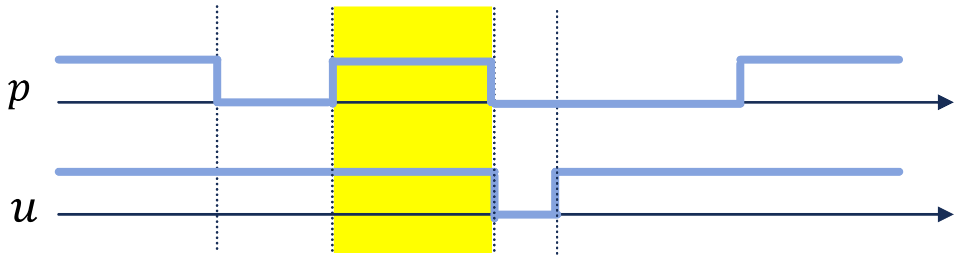
$$u = eventually_{[0,a]}\ p$$



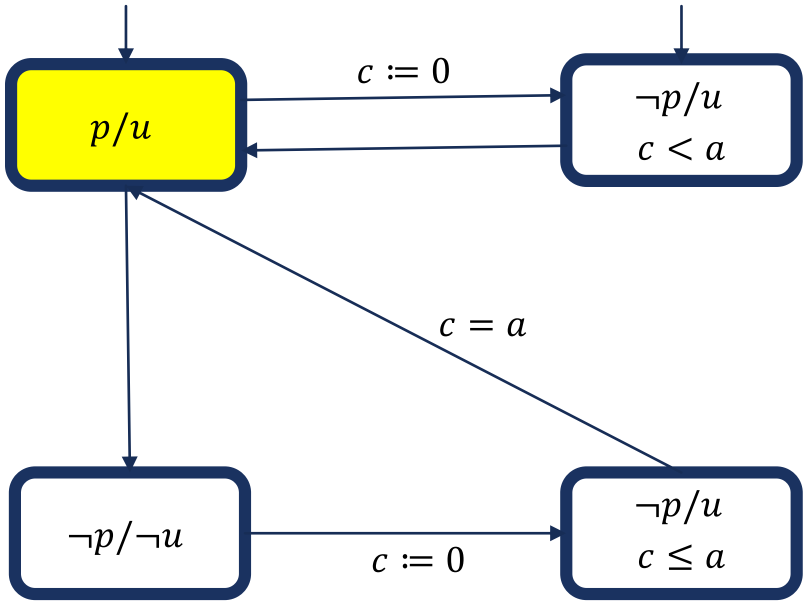
TEMPORAL TESTER FOR EVENTUALLY

Intuition

- Assume
  - Signals with left-closed right-open intervals only
  - Formula of the form $eventually_{[0,a]}\ p$



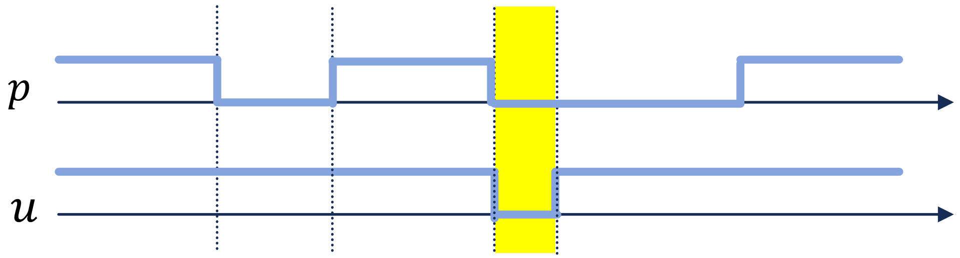
$$u = eventually_{[0,a]}\ p$$



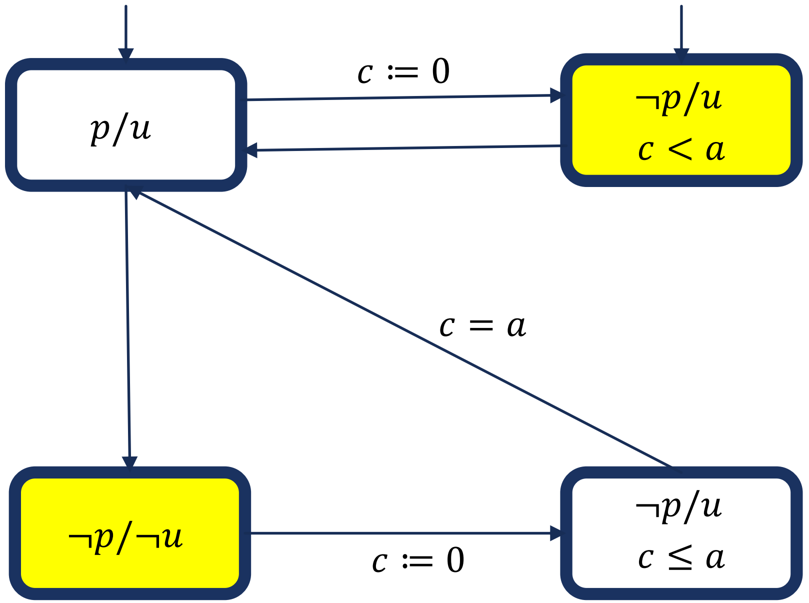
TEMPORAL TESTER FOR EVENTUALLY

Intuition

- Assume
  - Signals with left-closed right-open intervals only
  - Formula of the form $eventually_{[0,a]}\ p$



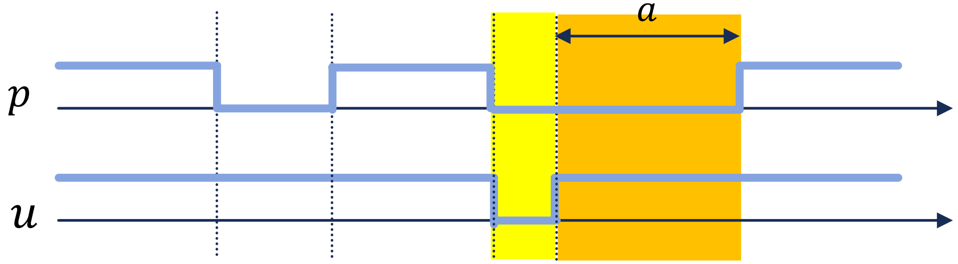
$$u = eventually_{[0,a]}\ p$$



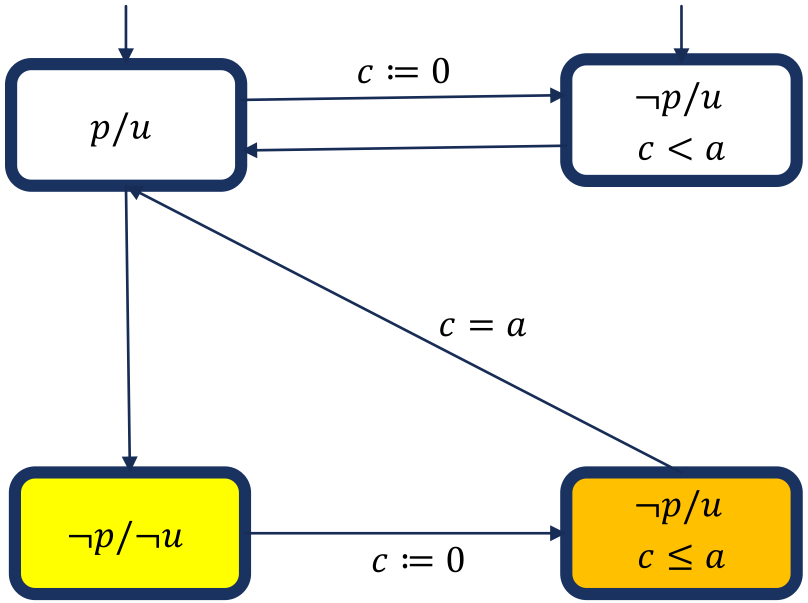
TEMPORAL TESTER FOR EVENTUALLY

Intuition

- Assume
  - Signals with left-closed right-open intervals only
  - Formula of the form $eventually_{[0,a]}\ p$



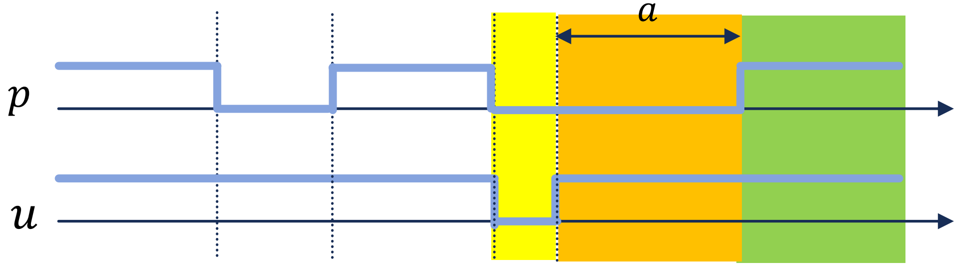
$$u = eventually_{[0,a]}\ p$$



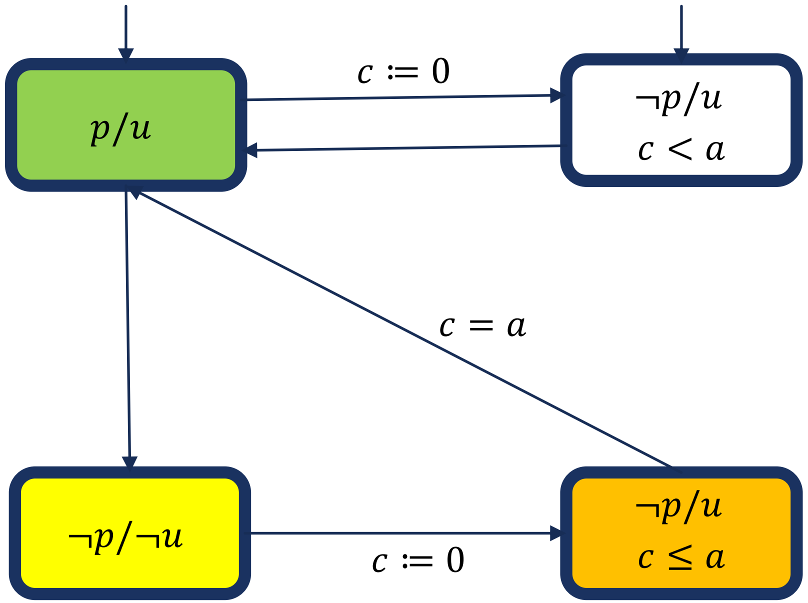
TEMPORAL TESTER FOR EVENTUALLY

Intuition

- Assume
  - Signals with left-closed right-open intervals only
  - Formula of the form $eventually_{[0,a]}\ p$



$u = eventually_{[0,a]}\ p$

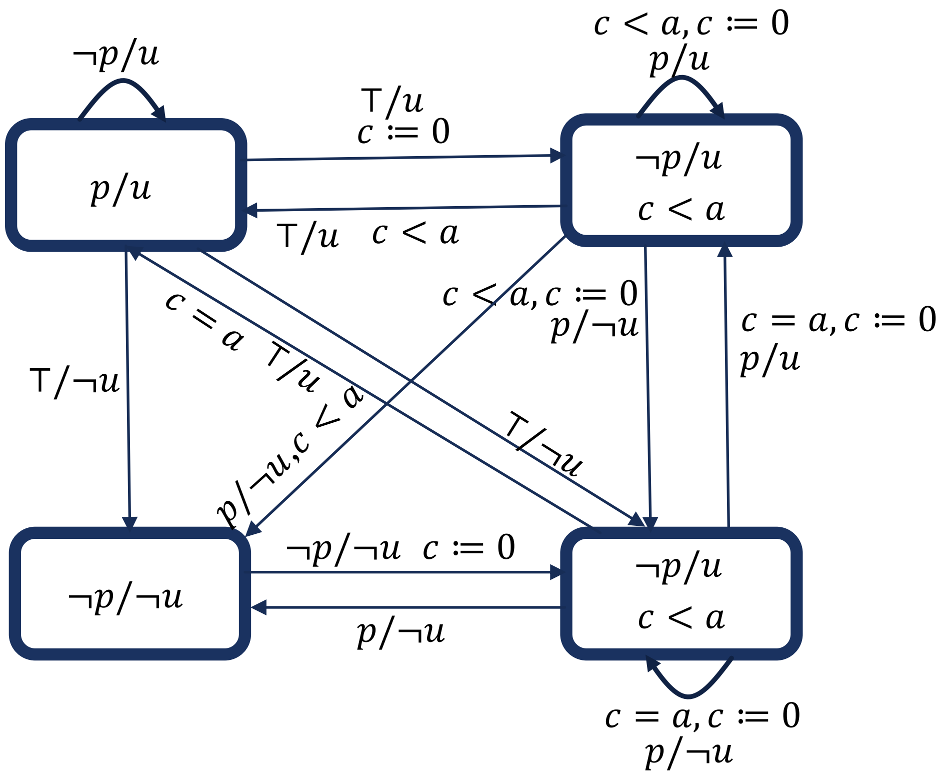


TEMPORAL TESTER FOR EVENTUALLY

Full construction

- Signals with arbitrary intervals
- Formula of the form $eventually_{(0,a)}\ p$

$$u = eventually_{(0,a)}\ p$$



CONCLUSIONS

- Details
  - Thomas Ferrère, Oded Maler, Dejan Nickovic, Amir Pnueli: From Real-time Logic to Timed Automata. J. ACM 66(3): 19:1-19:31 (2019)
- MITL can be fully expressed with two (simpler than U_I) temporal operators
  - U and $F_{(0,a)}$
- We provide a construction of timed testers for these operators
  - Straight-forward to understand and implement
  - At most 4 locations and 1 clock per tester
- Network of communicating testers, derived from the structure of an MITL property, yields an equivalent timed automaton
- Practical applications
  - Model checking, runtime verification, coverage-based testing, …

THANK YOU FOR YOUR ATTENTION