A Lattice-Theoretic Characterization of Safety and Liveness

Pete Manolios

PODC 2003

Joint work with Richard Trefler

University of Waterloo

Georgia Tech –Formal Methods Class- April 2004

Safety & Liveness

- Lamport classified program properties as:
  - Safety: nothing bad ever happens.
  - Liveness: something good eventually happens.
  - Neither: neither of the above.
- Transformational systems:
  - Safety: type/stack/memory safety.
    no reachable structures are deallocated.
    partial correctness.
  - Liveness: termination.
    unreachable structures are deallocated eventually.
  - Neither: total correctness.

Reactive Systems

- S/L used classify properties of *reactive* systems.
  - Distributed, concurrent systems engaged in ongoing behavior.
  - Examples: network protocols, pipeline machines, distributed systems, embedded systems, etc.
- Safety.
  - Only one process is in its critical section at any point in time.
  - Transactions appear to be atomic.
  - Messages are authenticated.
- Liveness.
  - Requests are eventually processed.
  - Weak/strong fairness (eventually always/ infinitely often).

Overview of Previous Work

- Specification: partial/total correctness, fairness, etc.
- Formal, topological characterization and decomposition theorem of Alpern & Schneider for linear time (semantic, ω-regular).
- Sistla's syntactic characterization for linear time temporal logics.
- Manolios and Trefler extension to branching time (subsumes linear time; includes process algebra, CTL, CTL*, μ-calculus).
- Different proof methods employed for safety & liveness: proofs by induction vs. construction of well-founded relations.
- In some cases there are decision procedures for safety properties, but not for liveness properties.
- In the context of model checking Kupferman & Vardi show that model checking safety is easier.
- Security: Schneider argues that *enforceable* security properties correspond to safety properties & security automata to Büchi automata. ...

Lattice Theoretic Approach

- Simpler and more general characterization.
  - Carefully analyzed conditions required to prove decomposition & related theorems.
  - Led to a lattice theoretic approach, where basic results are in terms of complemented modular lattices.
  - Simpler to apply: fewer properties to check.
  - Applicable in more contexts (e.g., closure operators are not required to distribute over joins).
- Allow us to simplify and unify previous results.
  - Characterization, decomposition theorem for Büchi automata, the main results in [AS87] follow directly.
  - Similarly with the semantic branching time results and those based on Rabin tree automata [MT01].

Outline

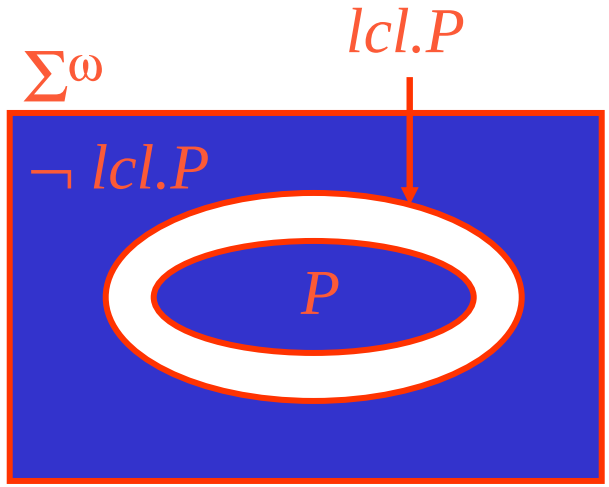
- Linear Time Framework
- Examples
- Büchi Automata
- Lattice Theoretic Characterization
- Branching time/Rabin automata
- Conclusions

Linear Time Framework

- Programs and properties are sets of ω-sequences.
- Alpern & Schneider give a topological characterization [AS85].
- *cl* is a *topological-closure* operator on X iff:
  - $cl.\emptyset = \emptyset$
  - $A \subseteq cl.A$
  - $cl(cl.A) = cl.A$
  - $cl(A \cup B) = cl.A \cup cl.B$
- A closure operator, *lcl*: $\mathcal{P}(\Sigma^\omega) \rightarrow \mathcal{P}(\Sigma^\omega)$, is defined as follows:
  $lcl.p = \{z \in \Sigma^\omega : \langle \forall x \in \Sigma^* : x \preccurlyeq z : \langle \exists y \in p :: x \preccurlyeq y \rangle \rangle\}$
- Note that *lcl* is a topological-closure operator.
- *p* is safe if $lcl.p = p$, *e.g.*, Gq (q always holds).
- *p* is live if $lcl.p = \Sigma^\omega$, *e.g.*, Fq (q eventually holds).

Decomposition Theorem

- $(P \cup \neg lcl.P)$ is a liveness property.



Decomposition Theorem

- *(P ∪ ¬lcl.P)* is a liveness property.

  lcl(P ∪ ¬lcl.P)

  = { *lcl* distributes over ∪ }

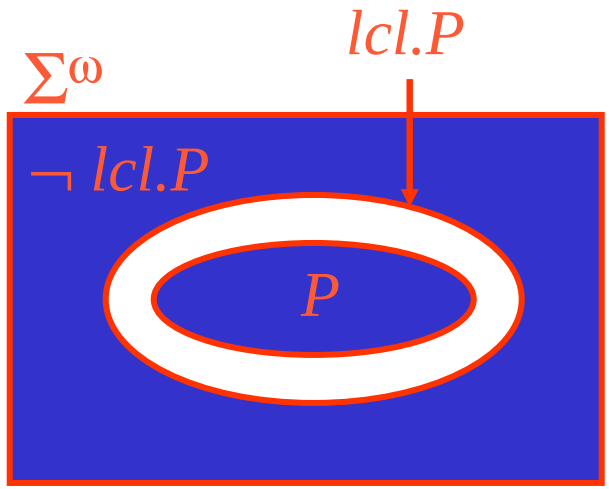
  lcl.P ∪ lcl(¬lcl.P)

  ⊇ { lcl.A ⊇ A }

  lcl.P ∪ ¬lcl.P

  = { Set theory }

  Σ^ω



Decomposition Theorem

- *(P ∪ ¬lcl.P)* is a liveness property.

  lcl(P ∪ ¬lcl.P)

  = { *lcl* distributes over ∪ }

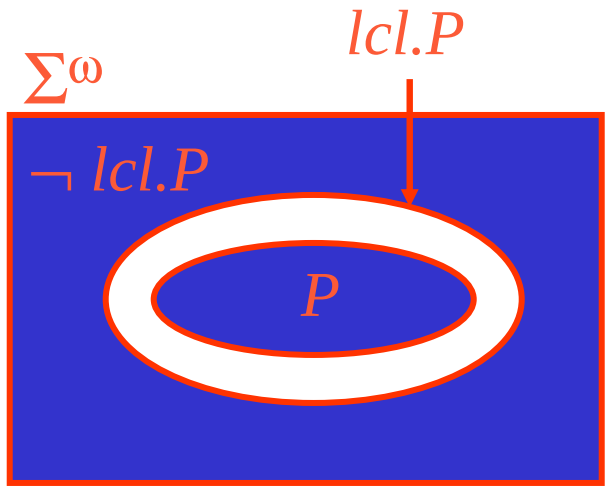
  lcl.P ∪ lcl(¬lcl.P)

  ⊇ { lcl.A ⊇ A }

  lcl.P ∪ ¬lcl.P

  = { Set theory }

  Σ^ω



- Any property is the intersection of a safe and live property.

  lcl.P ∩ (P ∪ ¬lcl.P) = (lcl.P ∩ P) ∪ (lcl.P ∩ ¬lcl.P) = P ∪ ∅ = P

Outline

- Linear Time Framework
- Examples
- Büchi Automata
- Lattice Theoretic Characterization
- Branching time/Rabin automata
- Conclusions

Examples

- P0: false (corresponds to $\emptyset$)
- P1: The first symbol is a
- P2: P1 and there is a non-a symbol
- P3: The number of a's is finite
- P4: a within n steps

Examples

P0: false (corresponds to $\emptyset$)	false
P1: The first symbol is a	a
P2: P1 and there is a non-a symbol	$a \wedge F\neg a$
P3: The number of a's is finite	$FG\neg a$
P4: a within n steps	$X^{\leq n}\, a$

Examples

- P0: false (corresponds to $\emptyset$) — false
- P1: The first symbol is a — a
- P2: P1 and there is a non-a symbol — $a \wedge F\neg a$
- P3: The number of a's is finite — $FG\neg a$
- P4: a within n steps — $X^{\leq n}\, a$

- Note that P0, P1, and P4 are safety properties.
- The closure of P2 is P1, so it is not a safety property; neither is it a liveness property.
- The closure of P3 is Σ^{ω} so it is a liveness property.

Outline

- Linear Time Framework
- Examples
- Büchi Automata
- Lattice Theoretic Characterization
- Branching time/Rabin automata
- Conclusions

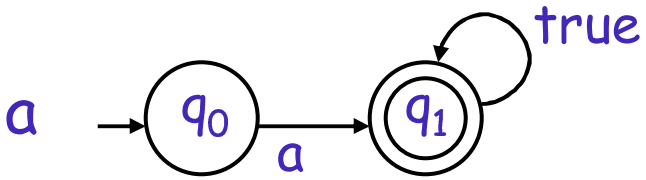
Büchi Automata

- Büchi automata recognize regular languages over ω-sequences.
- A Büchi automaton B is a tuple $\langle \Sigma, Q, q_0, \delta, F \rangle$ where:
  - Σ is a finite alphabet
  - Q is a finite set of states
  - q_0 is the start state
  - $\delta : Q \times \Sigma \to \mathcal{P}(Q)$ is the transition relation (notice non-deterministic)
  - F is a set of accepting states
- r is a <u>run</u> of t ($\in \Sigma^\omega$) on B if r is a Q-labeled sequence such that:
  - $r.0$ is labeled by $q0$
  - For all $i \in \omega$, $r(i+1) \in \delta(r.i, t.i)$
- A run is <u>accepting</u> iff some state of F occurs infinitely often in r.
- $\mathcal{L}.B = \{t :$ there is an accepting run of t on $B\}$.
- [AS87] shows $\langle \forall B :: \langle \exists B_S, B_L :: \mathcal{L}.B = \mathcal{L}.B_S \cap \mathcal{L}.B_L \rangle \rangle$ . . .

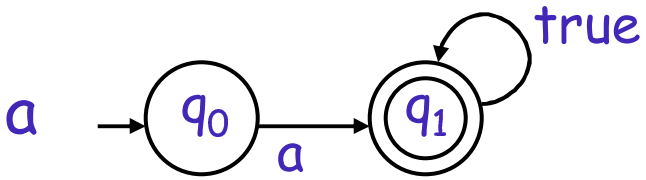
$$\langle \forall B :: \langle \exists B_S, B_L :: \mathcal{L}.B = \mathcal{L}.B_S \cap \mathcal{L}.B_L \rangle \rangle$$

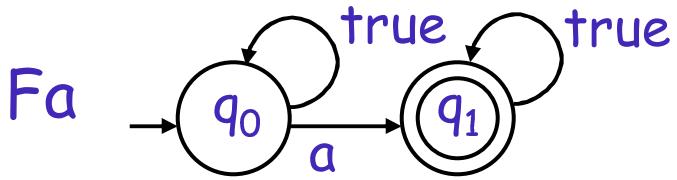
- The idea is to define a closure operator on Büchi automata
  - The operator removes states that cannot reach an accepting state.
  - It then makes every state accepting.
  - In this way, the fairness condition is made trivial.
- It can be shown that applying the operator to *B* results in an automaton whose language is the */c/* of the language of *B*.
- Since Büchi automata are closed under complementation, union, the liveness automaton is as before (decomposition theorem).
- This is the main result in [AS87].
- It required different proofs than those in [AS85] as Büchi automata do not define a topology.
- Model checking LTL: Turn into Büchi automaton, negate, intersect with the automaton for the Kripke structure and check for non-emptiness.
- Büchi automata are more expressive than LTL and, for some, easier to use and understand, so they are used to specify properties.

Büchi Automata Examples

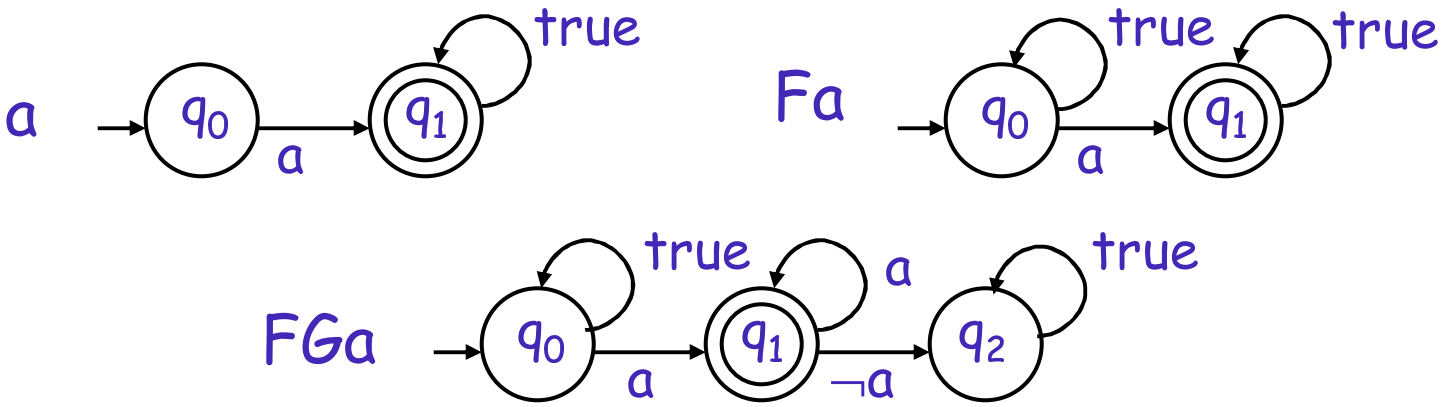


Büchi Automata Examples

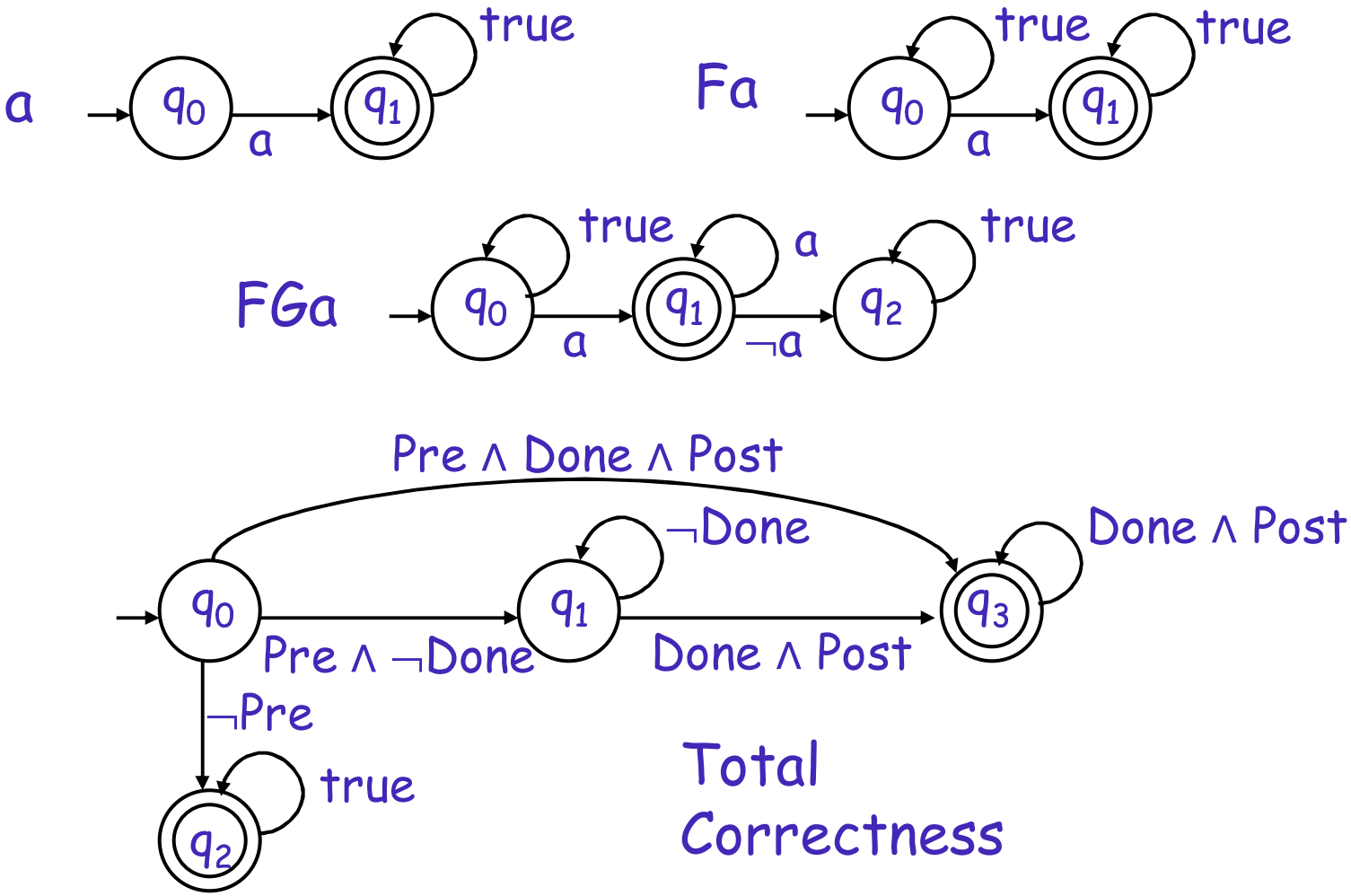




Büchi Automata Examples



Büchi Automata Examples



Outline

- Linear Time Framework
- Examples
- Büchi Automata
- Lattice Theoretic Characterization
- Branching time/Rabin automata
- Conclusions

Lattice Theory

- A *lattice* is a poset $\langle L, \preccurlyeq \rangle$ such that every pair has a glb, a meet ($\wedge$) and a lub, a join ($\vee$).
- Equivalently, a lattice is a triple $\langle L, \wedge, \vee \rangle$ such that:
  - $(a \vee b) \vee c = a \vee (b \vee c)$ (associative law)
  - $a \vee b = b \vee a$ (commutative law)
  - $a \vee a = a$ (idempotency law)
  - $a \vee (a \wedge b) = a$ (absorption law)
  - Each law also has a dual (interchange $\wedge$, $\vee$).
  - We define $a \preccurlyeq b \equiv (a \wedge b) = a$ (thus, $a \preccurlyeq b \equiv (a \vee b) = b$).

Lattice Theory

- A *lattice* is a poset $\langle L, \preccurlyeq \rangle$ such that every pair has a glb, a meet ($\wedge$) and a lub, a join ($\vee$).
- Equivalently, a lattice is a triple $\langle L, \wedge, \vee \rangle$ such that:
  - $(a \vee b) \vee c = a \vee (b \vee c)$ (associative law)
  - $a \vee b = b \vee a$ (commutative law)
  - $a \vee a = a$ (idempotency law)
  - $a \vee (a \wedge b) = a$ (absorption law)
  - Each law also has a dual (interchange $\wedge$, $\vee$).
  - We define $a \preccurlyeq b \equiv (a \wedge b) = a$ (thus, $a \preccurlyeq b \equiv (a \vee b) = b$).
- Lemma 1: (1) $a \preccurlyeq b \Rightarrow a \vee c \preccurlyeq b \vee c$

  (2) $a \preccurlyeq b \Rightarrow a \wedge c \preccurlyeq b \wedge c$

$a \vee c \preccurlyeq \{\text{Absorption } (x \preccurlyeq x \vee y)\}\ a \vee c \vee b = \{a \preccurlyeq b\}\ b \vee c$

Lattice Theory

- *A lattice-closure* on L is a function $cl : L \to L$ s.t.
  (1) $a \preccurlyeq cl.a$ (2) $cl.a = cl(cl.a)$ (3) $a \preccurlyeq b \Rightarrow cl.a \preccurlyeq cl.b$
- Lemma 2: $cl(a \vee b) \succcurlyeq cl.a \vee cl.b$
  $cl(a \vee b) = cl(a \vee b) \vee cl(a \vee b) \succcurlyeq\{a \vee b \succcurlyeq a, cl, \text{L1}\} cl.a \vee cl.b$

Lattice Theory

- A *lattice-closure* on L is a function $cl : L \to L$ s.t.
  (1) $a \preccurlyeq cl.a$ (2) $cl.a = cl(cl.a)$ (3) $a \preccurlyeq b \Rightarrow cl.a \preccurlyeq cl.b$
- Lemma 2: $cl(a \vee b) \succcurlyeq cl.a \vee cl.b$
  $cl(a \vee b) = cl(a \vee b) \vee cl(a \vee b) \succcurlyeq\{a \vee b \succcurlyeq a,\ cl,\ L1\}\ cl.a \vee cl.b$
- A lattice has a *unit* element, 1, if $\langle \forall a \in L :: a \wedge 1 = a \rangle$.
- A lattice has a *zero* element, 0, if $\langle \forall a \in L :: a \vee 0 = a \rangle$.
- If L has 1, 0, then b is a *complement* of a $(b \in cmp.a)$
  iff $b \wedge a = 0$ and $b \vee a = 1$.
- *Complemented lattice*: every element has a complement.
- A lattice is *modular* iff $a \succcurlyeq c \Rightarrow a \wedge (b \vee c) = (a \wedge b) \vee (a \wedge c)$
- Henceforth, $\langle L, \wedge, \vee, 0, 1 \rangle$ is a complemented, modular lattice.
- Notice that a Boolean algebra is a special case.

Decomposition Theorem

- A *cl-safety* property is one where $cl.a = a$.
- A *cl-liveness* property is one where $cl.a = 1$.
- Lemma: If $b \in cmp(cl.a)$ then $a \vee b$ is a cl-liveness element.
- Theorem: Every element is the meet of a *cl*-safety and *cl*-liveness element.

  Let $b \in cmp(cl.a)$

  $cl.a \wedge (a \vee b)$

  $= \{cl.a \geqslant a,$ Modularity$\}$

  $(cl.a \wedge a) \vee (cl.a \wedge b)$

  $= \{ a \leqslant cl.a \}$

  $a \vee (cl.a \wedge b)$

  $= \{ c \in cmp(b) \wedge a \leqslant b \Rightarrow a \wedge c = 0 \}$

  a

Decomposition Theorem

- A *cl-safety* property is one where $cl.a = a$.
- A *cl-liveness* property is one where $cl.a = 1$.
- Lemma: If $b \in cmp(cl.a)$ then $a \vee b$ is a cl-liveness element.
- Theorem: Every element is the meet of a *cl*-safety and *cl*-liveness element.

Let $b \in cmp(cl.a)$

$cl.a \wedge (a \vee b)$

$= \{cl.a \geqslant a,$ Modularity$\}$

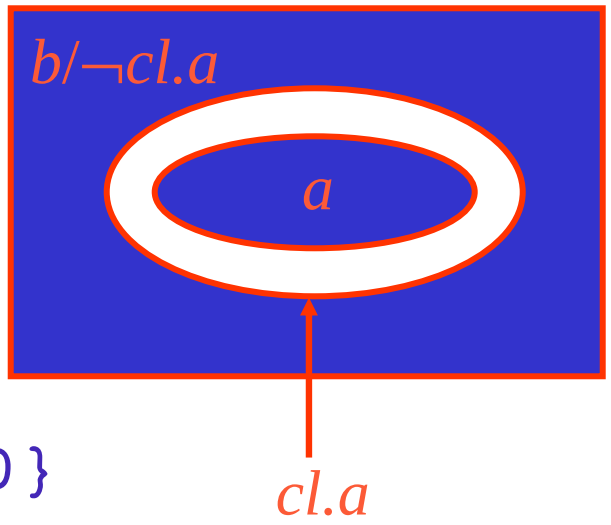
$(cl.a \wedge a) \vee (cl.a \wedge b)$

$= \{ a \preccurlyeq cl.a \}$

$a \vee (cl.a \wedge b)$

$= \{ c \in cmp(b) \wedge a \preccurlyeq b \Rightarrow a \wedge c = 0 \}$

a



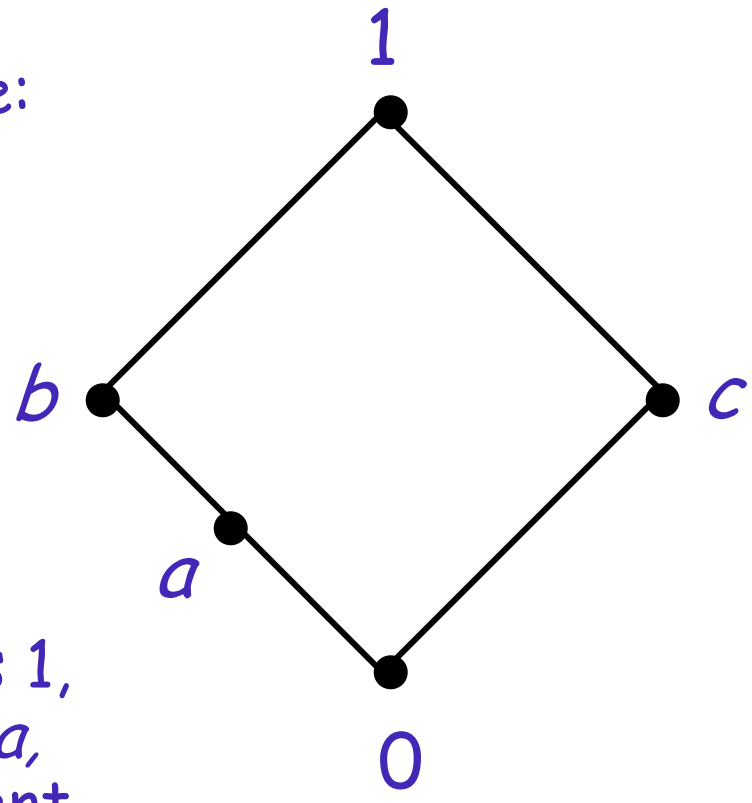
Why is modularity needed?

This is not a modular lattice:

- $b \gtrdot a$
- $b \wedge (c \vee a) = b$
- $(b \wedge c) \vee (b \wedge a) = a$

Define $cl.a = b$, identity otherwise.

The only liveness element is 1, so we cannot decompose a, as a is not a safety element.



Applications

- Corollary: Alpern & Schneider semantic results [AS85].

  $\langle \mathcal{P}(\Sigma^\omega), \cap, \cup, \varnothing, \Sigma^\omega, \neg \rangle$ is a Boolean algebra and *lcl* is a lattice-closure operator.
- Corollary: Alpern & Schneider Büchi automata results.

  Since Büchi automata are closed under union, intersection, and complementation, they form a Boolean algebra.

  Since the Büchi closure operator corresponds to *lcl*, we get the main results in [AS87].
- To get these results, we had to think syntactically:

  What properties did our proofs depend on?

Outline

- Linear Time Framework
- Examples
- Büchi Automata
- Lattice Theoretic Characterization
- Branching time/Rabin automata
- Conclusions

Branching Time Framework

- Programs and properties are sets of infinite trees.
- Due to the path quantifiers A and E.
- We distinguish between A and E.
  - AGq: along all paths Gq (universally safe).
  - EGq: along some path Gq (existentially safe).
- The branching framework is important because:
  - It is used in process algebra.
  - Model checking tools, *e.g.*, SMV and VIS, are based on CTL, a branching time logic.

Trees

- A tree, t, is a prefix-closed subset of $\mathbb{N}^*$, labeled from Σ
- $t \in A^{tot}$ (is total) if $t \neq \emptyset$ and $\langle \forall x \in t :: \langle \exists y \in t :: x \prec y \rangle \rangle$
- $t \in A^{f}$ (is finite-depth) if $\langle \exists n \in \mathbb{N} :: \langle \forall s \in t :: \#s < n \rangle \rangle$
- A^{nt}: the set of non-total trees
- A^{all}: $A^{tot} \cup A^{nt}$
- $\sqsubseteq$: a partial order on trees analogous to $\preccurlyeq$ for sequences
- $x \sqsubseteq y$ if y can be obtained by extending x at its leafs

Closures

- $lcl.p = \{z \in \Sigma^{\omega} : \langle \forall x \in \Sigma^{*} : x \preccurlyeq z : \langle \exists y \in p :: x \preccurlyeq y \rangle \rangle\}$
- $ncl.p = \{z \in A^{tot} : \langle \forall x \in A^{nt} : x \sqsubseteq z : \langle \exists y \in p :: x \sqsubseteq y \rangle \rangle\}$
- $fcl.p = \{z \in A^{tot} : \langle \forall x \in A^{f} : x \sqsubseteq z : \langle \exists y \in p :: x \sqsubseteq y \rangle \rangle\}$

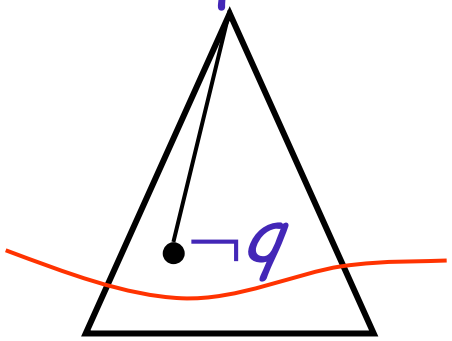
Properties of *ncl* and *fcl*

- $p \subseteq ncl.p, \quad p \subseteq fcl.p$
- $ncl(ncl.p) = ncl.p, \quad fcl(fcl.p) = fcl.p$
- $fcl(p \cup s) = fcl.p \cup fcl.s, \quad ncl(p \cup s) \supseteq ncl.p \cup ncl.s$
- $ncl.p \subseteq fcl.p$
- *fcl* defines a topology; *ncl* does not
  ($ncl(p \cup s) \subseteq ncl.p \cup ncl.s$ does not hold)

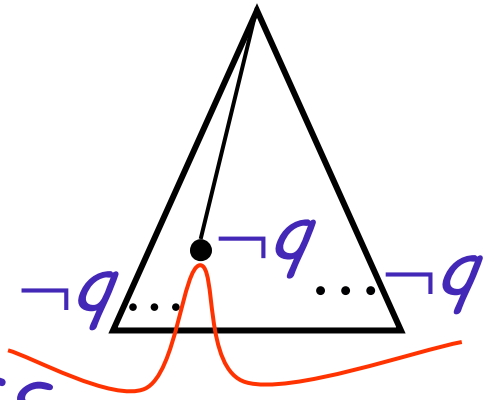
Safety

- $p \in$ US (is universally safe) if $p = fcl.p$
- $p \in$ ES (is existentially safe) if $p = ncl.p$

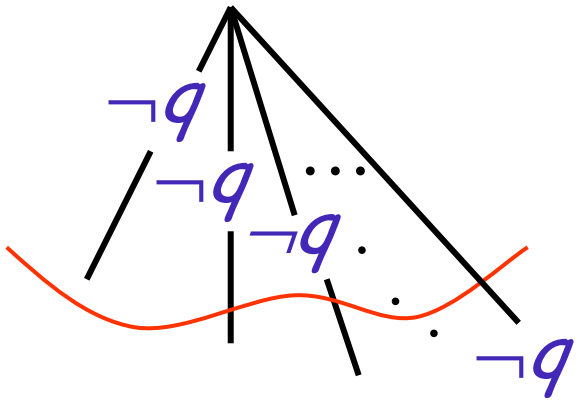
$AGq \in$ US



$EGq \in$ ES



$EGq \notin$ US

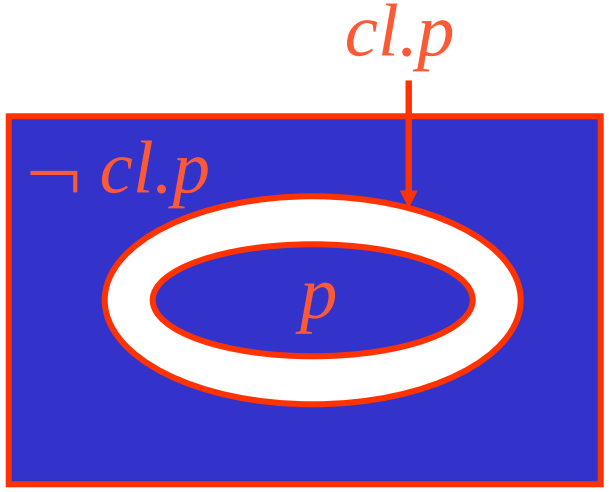


Also, $AGq \in$ ES,
as ncl.p $\subseteq$ fcl.p

Liveness

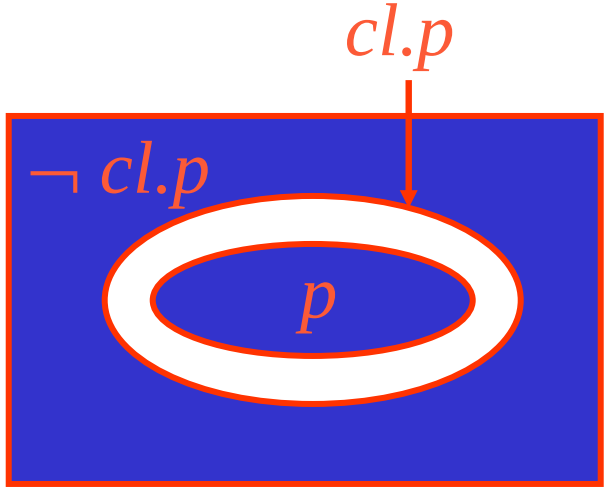
- $p \in$ UL (is universally live) if $fcl.p = A^{tot}$
- $p \in$ EL (is existentially live) if $ncl.p = A^{tot}$

- AFq $\in$ UL
- EFq $\in$ EL
- $[p \cup \neg(fcl.p)] \in$ UL
- $[p \cup \neg(ncl.p)] \in$ EL



Decomposition Theorems

- Every property is the intersection of
  - A universally safe/ universally live property
  - An existentially safe/ existentially live property
  - An existentially safe/ universally live property
- For every property p
  - $p = fcl.p \cap [p \cup \neg(fcl.p)]$
  - $p = ncl.p \cap [p \cup \neg(ncl.p)]^*$
  - $p = ncl.p \cap [p \cup \neg(fcl.p)]$
- If s is safe and $s \cap l = p$, then $ncl.p \subseteq s$, $l \subseteq [p \cup \neg(ncl.p)]$



Regular Languages

- Rabin tree automata recognize regular languages of k-ary ω-trees.
- A Rabin automaton B is a tuple $\langle \Sigma, Q, q_0, \delta, \Phi \rangle$ where
  - Σ is a finite alphabet
  - Q is a finite set of states
  - q_0 is the start state
  - $\delta : Q \times \Sigma \to \mathcal{P}(Q^k)$ is the transition relation
  - Φ is the acceptance condition

Regular Languages

- r is a run of t on B if r is a k-ary tree whose
  - root is labeled by $q0$
  - For every node σ in r with successors labeled $q_1,\ldots,q_k$, $(q_1,\ldots,q_k)$ in $\delta(r.\sigma, t.\sigma)$
- A run is accepting iff all infinite paths satisfy Φ
- Φ is a set of pairs $(green.i, red.i) \in (\mathcal{P}.Q)^2$, $[1..m]$
- $\Phi = \vee_{i\in[1..m]}[(\vee_{g\in green.i} \mathrm{GF}\ g) \wedge (\wedge_{r\in red.i} \mathrm{FG}\ \neg r)]$
- $\mathcal{L}.B = \{t$: there is an accepting run of B on $t\}$

Decomposition Theorems

- For any Rabin automaton, B, there exist effectively derivable automata B_s and B_l such that $\mathcal{L}.B = \mathcal{L}.B_s \cap \mathcal{L}.B_l$, $\mathcal{L}.B_s$ is safe, and $\mathcal{L}.B_l$ is live.
- As before, B_s and B_l can be
  - Universally safe/ universally live
  - Existentially safe/ existentially live
  - Existentially safe/ universally live

Conclusions

- Lattice theoretic characterization of safety and liveness.
- Applications to linear time and branching time frameworks.
- The key was to formalize and then study the syntactic properties of the proof.
- Future Directions.
  - Define subclasses of safety and liveness formulas.
  - Syntactically characterize safety and liveness in branching time (Sistla has done both for the linear time case).
  - Is the model checking problem simpler? (Kupferman and Vardi have looked at this, mostly for linear time)
  - Any consequences for security automata?