Model Checking Collaboration, Competition and Dense Fault Resilience

研究生:黃重豪
指導教授:王凡 博士
口試委員:
地點:
時間:

Outline

- Motivation
- Contribution
- Game graph
- BSIL & TCL
  - Existing logics about game and strategy
  - Running example
  - Syntax and semantics
  - Expressive power
  - Algorithm and complexity
- SW Resilience against Dense Errors
  - Fault tolerance
  - Safety resilience games
  - AMC with events
  - Algorithm and complexity
- Experiment
- Conclusion

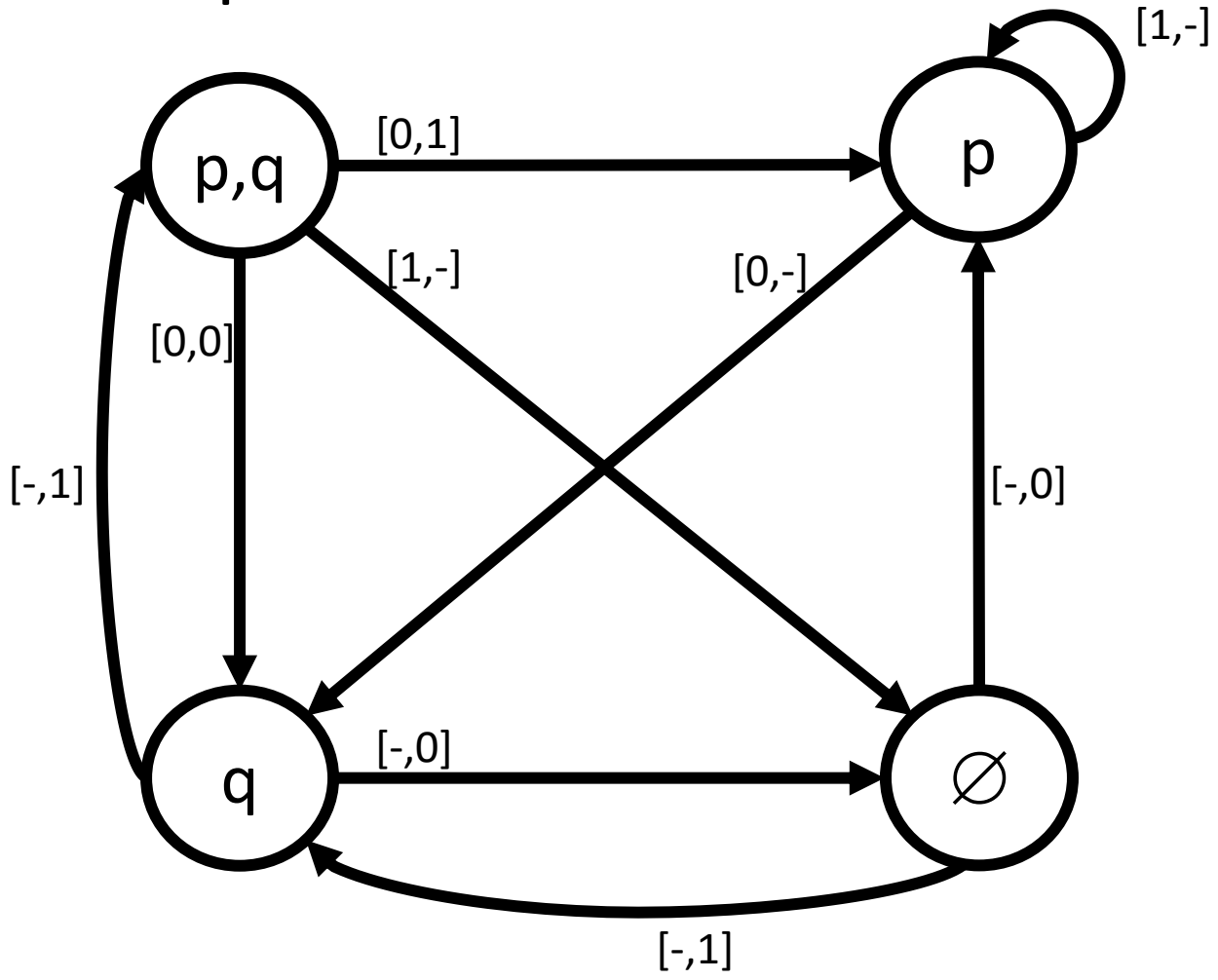
Motivation

- Formal verification of open systems requires taking user's behavior into consideration.
- Existing techniques suffer from either lack of expressive power or expensive time complexity.
- If generic model checking techniques are unavoidably with high complexity, we can try to focus on certain aspect and develop corresponding algorithm to solve the problem.

Contribution

- Logics, BSIL&TCL, which can specify the relationship between the strategies used to fulfill different sub-formula
  - Syntax & semantics
  - Expressive power
  - Model checking algorithm
  - Complexity
- A new criteria called dense fault resilience
  - Addressing the justification
  - An algorithm to verify the dense fault resilience of open systems
  - The complexity proof of the algorithm
- Implementation and experiment

Game Graph



Game Graph

- concurrent game graph (CGG) is a tuple $A = \langle m, Q, r, P, \lambda, E1, E2, \Delta, \delta \rangle$
  - M is the number of agents in the game.
  - Q is a finite set of states.
  - $r \in Q$ is the *initial state* of A.
  - P is a finite set of atomic propositions.
  - Function $\lambda : Q \rightarrow 2P$ labels each state in Q with a set of atomic propositions.
  - E1 and E2 are finite sets of move symbols that the protagonist and the antagonist can respectively choose in transitions. A pair in E1xE2 is called a move vector.
  - E2 is partitioned into error and and non-error moves (Eerror and Enoerr)
  - Δ is a set of tokens that can be issued by the agents during transitions.
  - $\delta : (R \times [1,m]) \rightarrow$ is a function that specifies the token (move symbol) issued by each agent in a transition.

Existing logics about game and strategy

- ATL* [AHK2002]
  - <1, 2, 3> ((◇ ¬j1)∧(◇ ¬j2)∧(◇ ¬j3))
- AMC (Alternating μ-Calculus) [AHK2002]
  - μ x.{1, 2, 3}(x ∨ ((◇ ¬j1)∧(◇ ¬j2)∧(◇ ¬j3)))
- GL (Game Logic) [AHK2002]
  - ∃{1,2,3} ((◇ ¬j1)∧(◇ ¬j2)∧(◇ ¬j3))
- SL (Strategy Logic) [CHP2010]
  - ⟨⟨x⟩⟩ ⟨⟨y⟩⟩ ⟨⟨z⟩⟩(1,x)(2,y)(3,z) $\wedge_{a\in[1,3]}$ ◇¬ j_a
  - ⟨⟨S1⟩⟩ ⟨⟨S2⟩⟩ ⟨⟨S3⟩⟩(1,S1)(((2,S2)(3,S3)((◇ ¬j1)∧(◇ ¬j2))∧((2,S3)(3,S3)◇ ¬j3))

Basic Strategy-Interaction Logic(BSIL)

- An extension of ATL
- Can specify the relationship between strategies used to satisfy different sub-formulas
- Relatively low model checking complexity(PSPACE)
- Ex.
  $\langle 1,2,3\rangle \wedge a \in [1,3] \Diamond \neg$ ja
  All player can cooperate to avoid getting into jail
  $\langle 1,2\rangle((\langle +\varnothing\rangle \Diamond \neg j_3) \wedge (\langle +3\rangle \Diamond \neg (j_1 \wedge j_2) \wedge (\langle +3\rangle \Box (j_1 \wedge j_2)))$
  Player 1 and player 2 have strategy to make player 3 keep free.
  And under the same strategy player 3 can make player 1 and player 2 free.
  And under the same strategy player 3 can make player 1 and player 2 always in jail.

Temporal Cooperation Logic(TCL)

- An extension of BSIL
- More expressive power
- Higher model checking complexity(EXPTIME)
- Ex.
  $\langle 1\rangle\Diamond((\langle +\rangle\bigcirc\neg betray_1)\wedge(betray_2 \vee betray_3))$
  Player 1 is forgiving
  $\langle 2\rangle((\langle +\rangle\Box\neg betray_2)$
  $\vee \langle +1\rangle\Diamond((\langle +\rangle\bigcirc\neg betray_1)\wedge(betray_2 \vee betray_3)))$
  Player 2 should avoid betrayal while player 1 can be unforgiving

Comparison

- Expressiveness:
  - BSIL&TCL can specify the interaction between the “strategies” while ATL*, GL, and AMC cannot
  - BSIL is less expressive than SL
- Model checking complexity:
  - ATL: PTIME
  - ATL*, GL, SL: Doubly EXPTIME-complete
  - AMC: EXPTIME-hard
  - BSIL: PSPACE-complete
  - TCL: EXPTIME-complete

Running Example

- To precisely express following spec for a bank system with 3 player(bank B, client C, and partner bank PB)
  - B has to make sure no one can check other user's password
  - B and C can cooperate to let C can deposit his account
  - B, C and PB can cooperate to let C transfer his money from PB to B
  - The previous 3 requirements share the same strategy of B
- ATL:
  $\langle B \rangle(\Box \neg checkPW \wedge \langle C \rangle \Diamond depositDone \wedge \langle C,PR \rangle \Diamond transferDone)$
  or
  $\langle B \rangle \Box \neg checkPW \wedge \langle B,C \rangle \Diamond depositDone \wedge \langle B,C,PR \rangle \Diamond transferDone$
  are both not fit the requirement perfectly

Running Example(cont.)

- SL:
  <*B1*><*C1*><*C2*><*PB1*>[*C3*][*PB2*][*PB3*](
  (B,*B1*)(C,*C3*)(PB,*PB2*)□¬checkPW
  ∧ (B,*B1*)(C,*C1*)(PB,*PB3*)◇depositDone
  ∧ (B,*B1*)(C,*C2*)(PB,*PB1*)◇transferDone)
  can perfectly describe the spec.
  But the model checking complexity is Doubly EXPTIME-complete
- BSIL:
  <B>(□ ¬checkPW ∧ <+C> ◇depositDone ∧ <+C,PR> ◇transferDone)

Syntax

- BSIL
  - state formulas $\varphi ::= p \mid \neg\varphi 1 \mid \varphi 1 \vee \varphi 2 \mid \langle A\rangle\tau \mid \langle A\rangle\vartheta$
  - tree formulas $\tau ::= \tau 1 \vee \tau 2 \mid \tau 1 \wedge \tau 2 \mid \langle +A\rangle\tau 1 \mid \langle +A\rangle\vartheta$
  - path formulas $\vartheta ::= \neg\vartheta 1 \mid \vartheta 1 \vee \vartheta 2 \mid \bigcirc \varphi 1 \mid \varphi 1 \mathrm{U} \varphi 2$
- TCL
  - state formulas $\varphi ::= p \mid \neg\varphi_1 \mid \varphi_1 \vee \varphi_2 \mid \langle A\rangle\tau$
  - tree formulas $\tau ::= \varphi \mid \tau_1 \vee \tau_2 \mid \tau_1 \wedge \tau_2 \mid \langle +A\rangle\bigcirc\tau_1 \mid \langle +A\rangle\vartheta_1 \mathrm{U}\,\tau_1 \mid \langle +A\rangle\vartheta_1 \mathrm{R}\,\tau_1 \mid \langle -A\rangle\bigcirc\tau_1 \mid \langle -A\rangle\vartheta_1 \mathrm{U}\,\tau_1 \mid \langle -A\rangle\vartheta_1 \mathrm{R}\,\tau_1$
  - path formulas $\vartheta ::= \varphi \mid \vartheta_1 \vee \vartheta_2 \mid \vartheta_1 \wedge \vartheta_2 \mid \langle +A\rangle\bigcirc\vartheta_1 \mid \langle +\rangle\vartheta_1 \mathrm{U}\,\vartheta_1 \mid \langle +\rangle\vartheta_1 \mathrm{R}\,\vartheta_1 \mid \langle -A\rangle\bigcirc\vartheta_1 \mid \langle -A\rangle\vartheta_1 \mathrm{U}\,\vartheta_1 \mid \langle -A\rangle\vartheta_1 \mathrm{R}\,\vartheta_1$
- No universal strategy quantifier [+A]

BSIL Semantics

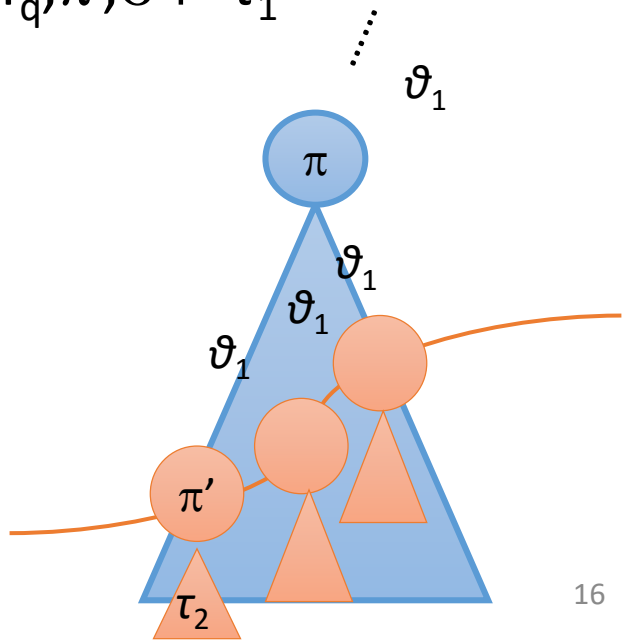
- $G, q \models_{\Sigma} p$ if and only if $p \in \lambda(q)$.
- For state formula $\varphi 1$, $G, q \models_{\Sigma} \neg\varphi 1$ if and only if $G, q \models_{\Sigma} \varphi 1$ is false.
- For state or tree formulas $\psi 1$ and $\psi 2$, $G, q \models_{\Sigma} \psi 1 \wedge \psi 2$ if and only if $G, q \models \psi 1$ and $G, q \models \psi 2$.
- For state or tree formulas $\psi 1$ and $\psi 2$, $G, q \models_{\Sigma} \psi 1 \vee \psi 2$ if and only if either $G, q \models_{\Sigma} \psi 1$ or $G, q \models_{\Sigma} \psi 2$.
- $G, q \models_{\Sigma} \langle A \rangle \tau$ if and only if there exists an S-profile $\prod$ of A with $G, q \models_{\prod} \tau$.
- G, q $\models_{\Sigma}$ <+A>τ if and only if there exists an S-profile $\prod$ of A with G, q $\models_{\Sigma \circ \prod}$ τ
- G, q $\models_{\Sigma}$ <A>θ if and only if there exists an S-profile $\prod$ of A such that, for all plays ρ from q compatible with $\prod$, ρ $\models_{\prod}$ θ holds.
- G, q $\models_{\Sigma}$ <+A>θ if and only if there exists an S-profile $\prod$ of A such that, for all plays ρ from q compatible with $\Sigma \circ \prod$, ρ $\models_{\Sigma \circ \prod}$ θ holds.

BSIL Semantics(cont.)

- For a path formula $\vartheta 1$, $\rho \models_{\Sigma} \neg\vartheta 1$ if and only if it is not the case that $\rho \models_{\Sigma} \vartheta 1$.
- For path formulas $\vartheta 1$ and $\vartheta 2$, $\rho \models_{\Sigma} \vartheta 1 \vee \vartheta 2$ if and only if either $\rho \models_{\Sigma} \vartheta 1$ or $\rho \models_{\Sigma} \vartheta 2$.
- $\rho \models_{\Sigma} \bigcirc \psi 1$ if and only if $G, \rho[1,\infty) \models_{\Sigma} \psi 1$.
- $\rho \models_{\Sigma} \psi 1 \mathrm{U} \psi 2$ if and only if there exists an $h \geq 0$ with $G, \rho[h,\infty) \models \psi 2$ and for all $j \in [0, h)$, $G, \rho[j,\infty) \models_{\Sigma} \psi 1$.

TCL Semantics

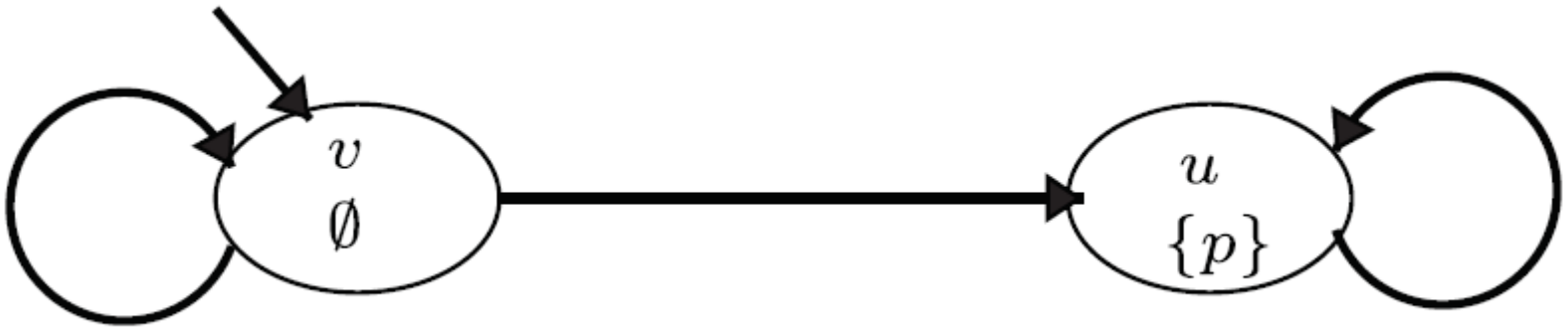
- State Formula
  - $T_q,\pi,\sigma \models \phi$ iff $last(\pi) \models \phi$
- Tree Formula
  - $T_q,\pi,\sigma \models <+A> \bigcirc \tau_1$ iff each successor π' of π in $T_q < \pi, \sigma, \psi >$, $T_q,\pi',\sigma \models \tau_1$
  - $T_q,\pi,\sigma \models <+A> \vartheta_1 U \tau_2$ iff $\exists k \geq |\pi|-1$ s.t
    $T_q,\pi'[0,k],\sigma \models \tau_2$ and
    $\forall h \in [|\pi|-1, k-1]$, $T_q,\pi'[0,h],\sigma \models \vartheta_1$



Memoryful

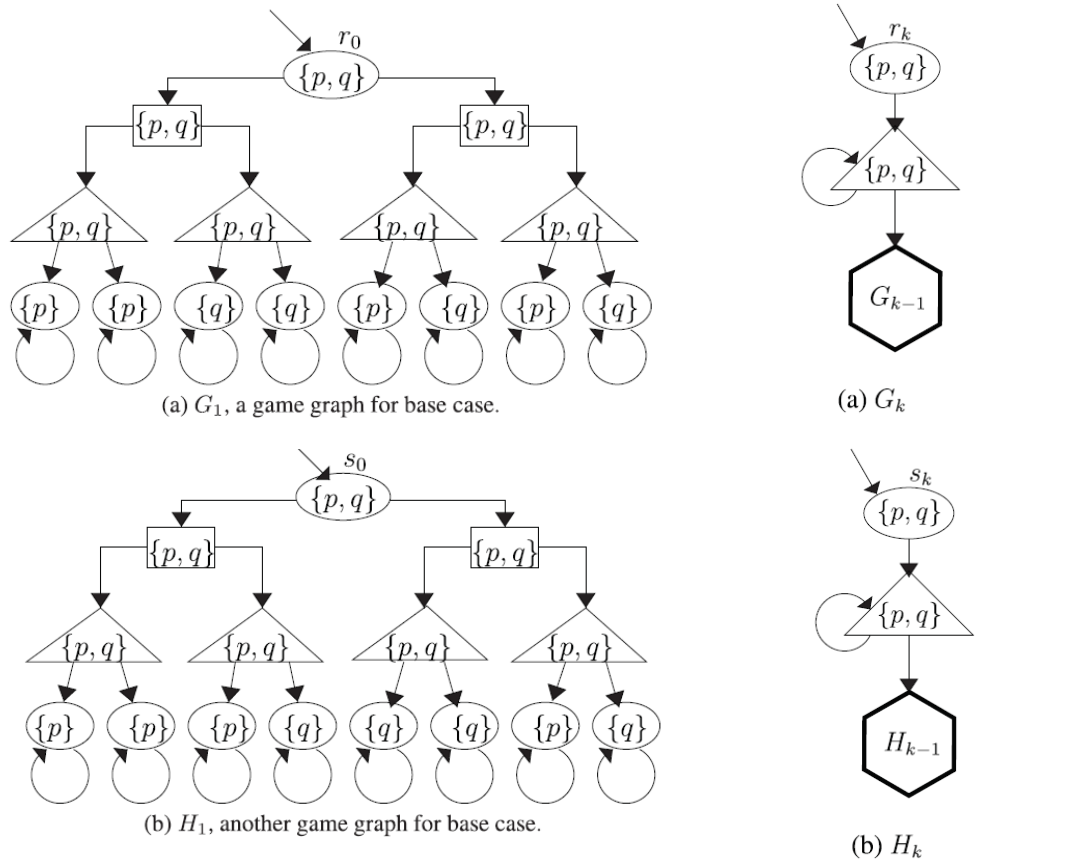
- The following is a formula in BSIL∩ATL* which cannot be fulfilled through a memoryless strategy

  <1>((¬○*p*)∧◇*p*)



Expressive power

- BSIL: <1>((<+2>□p)∧(<+2>□q))
- GL: with 1 modal operator
  let $\varphi 1$ be ∃ψ, ∀ψ
  $\exists\emptyset.\varphi 1$
  $\exists\{1\}.\varphi 1$, $\exists\{2\}.\varphi 1$, $\exists\{3\}.\varphi 1$
  $\exists\{1,2\}.\varphi 1$, $\exists\{2,3\}.\varphi 1$, $\exists\{1,3\}.\varphi 1$
  $\exists\{1,2,3\}.\varphi 1$
- GL with 1 modal operator cannot tell the difference between G1, H1
- GL with k modal operator cannot tell the difference between Gk, Hk



(a) G_1, a game graph for base case.

(a) G_k

(b) H_1, another game graph for base case.

(b) H_k

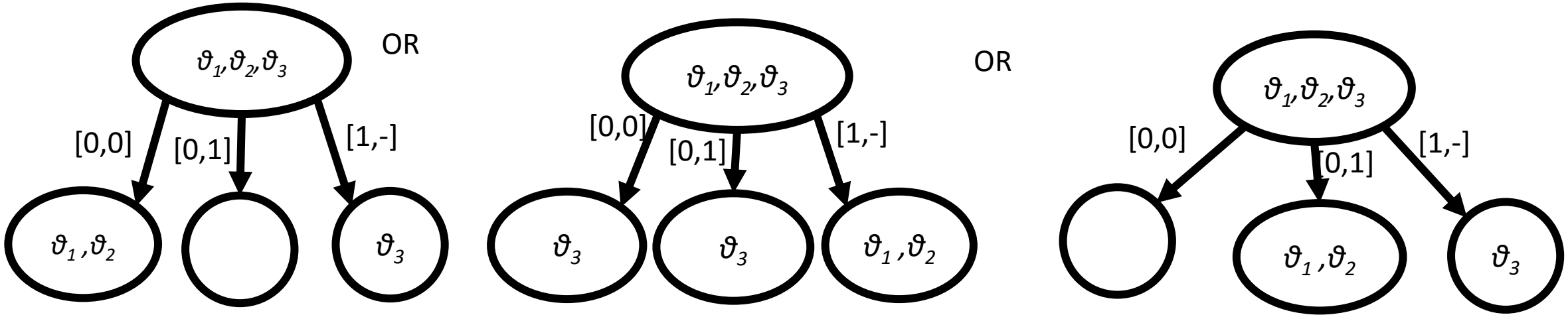
◯ belongs to Agent 1; □ belongs to Agent 2; and △ belongs to Agent 3.

Algorithm-BSIL to DNBB

- *disjunctive normal Boolean bound formula*
- BSIL:<1, 2>(<+3>(□*p* ∨ ◇*q*) ∧ <+3>(<+2>◇*r* ∨ <+4>□*q*))
- DNBB:
  ((1, *s*1)(2, *s*2)(3, *s*3)(□*p*∨◇*q*)∧(1, *s*1)(2, *s*5)(3, *s*4)◇*r*)
  ∨((1, *s*1)(2, *s*2)(3, *s*3)(□*p*∨◇*q*)∧(1, *s*1)(2, *s*2)(3, *s*4)(4, *s*6)□*q*)

Algorithm-Pass down obligation

- (1.a)$\vartheta_1 \wedge$ (1.a)$\vartheta_2 \wedge$ (1.b)ϑ_3

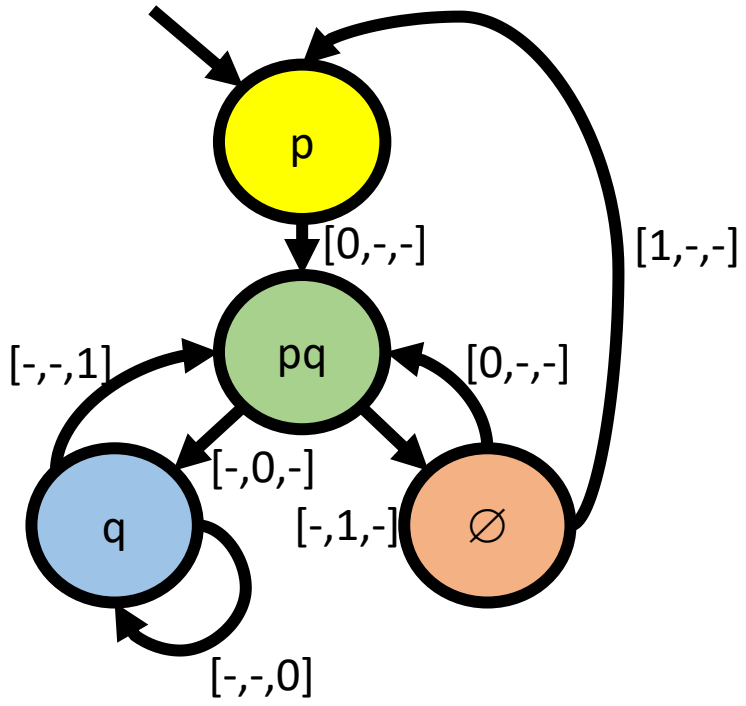


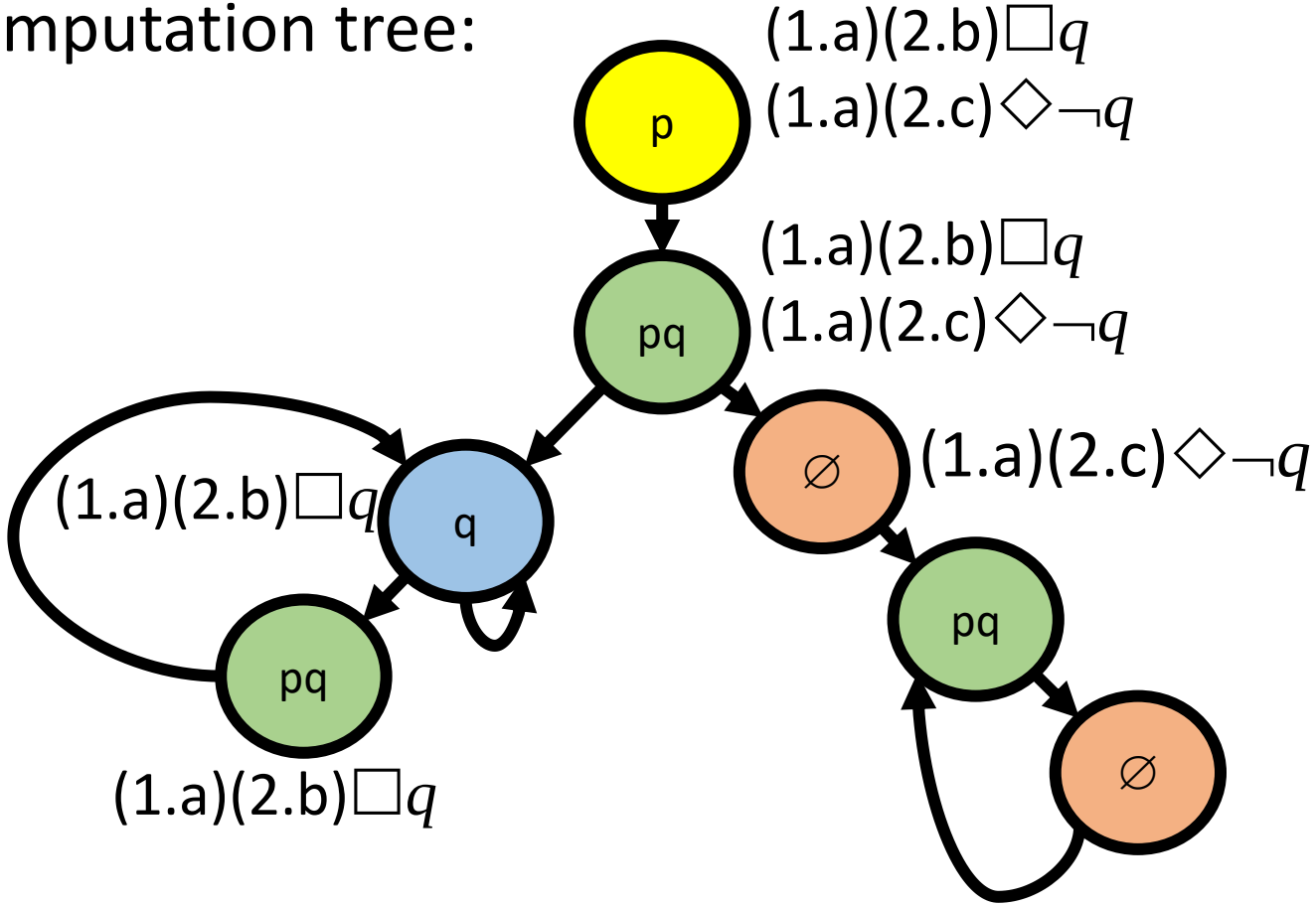
Algorithm-BSIL

- Rewrite BSIL to DNBB
- Guess the obligations
- Pass down the unfulfilled obligations according to the strategy variable binding and the transition function of current state
- Recursively check the successors in the computation tree
- While encountering repeated game state, if the obligation set remain the same, return false.

Model Checking Example-BSIL

- Checking (1.a)(2.b)$\Box q \wedge$(1.a)(2.c)$\Diamond \neg q$
- Game graph: Computation tree:



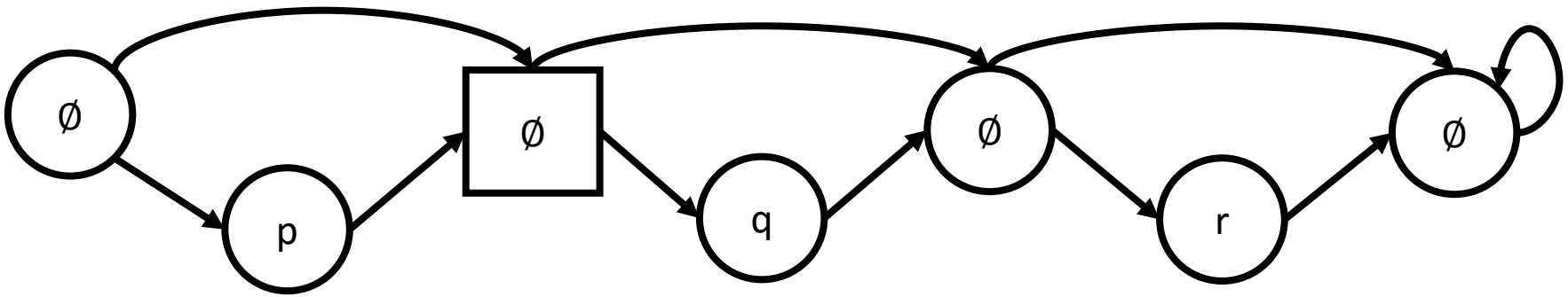


Complexity-BSIL model checking is PSPACE-easy

- Let |x| be the length of the BSIL formula
- The calculation and pass down of obligations can be done non-deterministically in linear time
- It takes at most |Q| steps to satisfy an UNTIL expression
- The maximum depth of the computation tree is |x||Q|

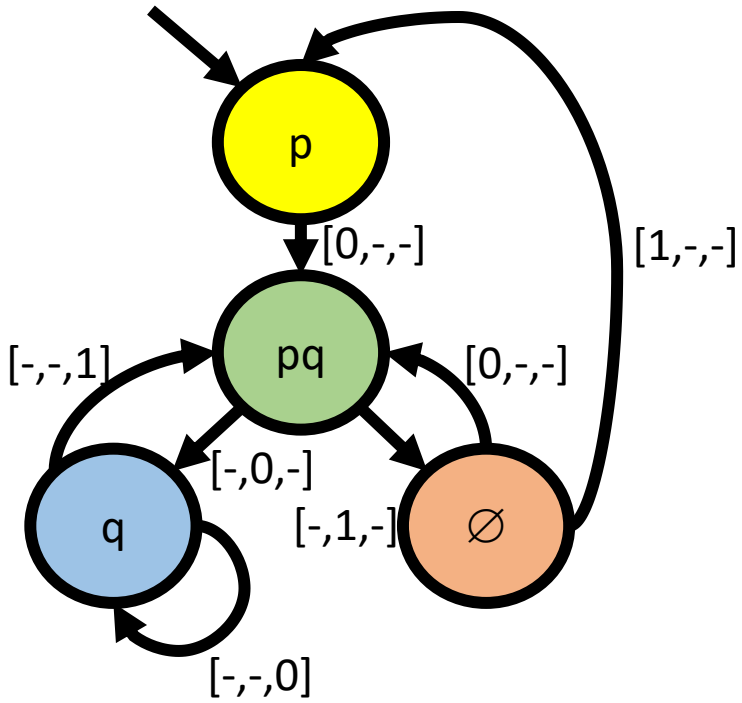
Complexity-BSIL model checking is PSPACE-hard

- Given QBF property η ≡ ∃p∀q∃r((p∨ q ∨ r) ∧ (¬p∨¬r))
- BSIL:<1>((◇p∨◇q∨◇r) ∧ (¬□p∨¬□r))

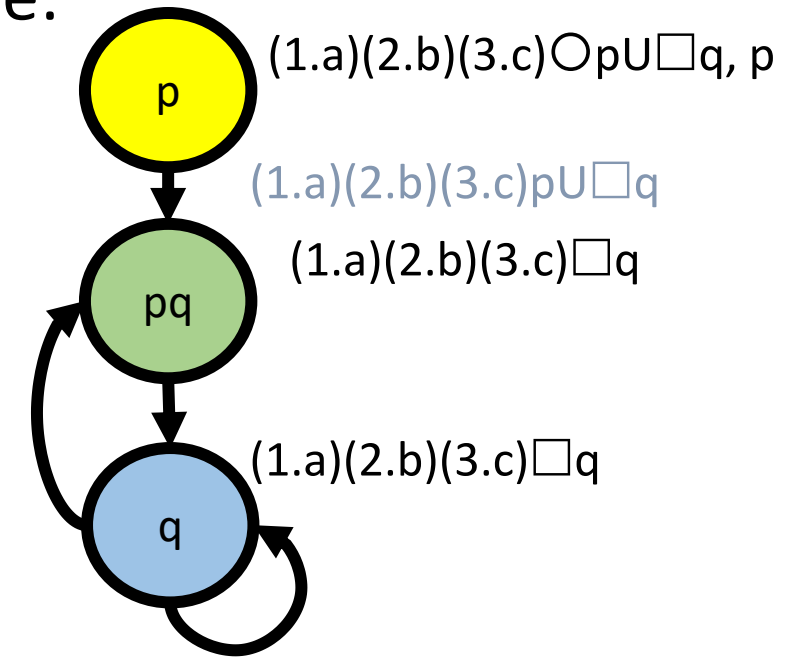


Model Checking Example-TCL

- Checking <1>(<+2>pU<+3>□q)
- Closure:{p, q, pU□q, □q, ○pU□q}
- Game graph:



Computation tree:

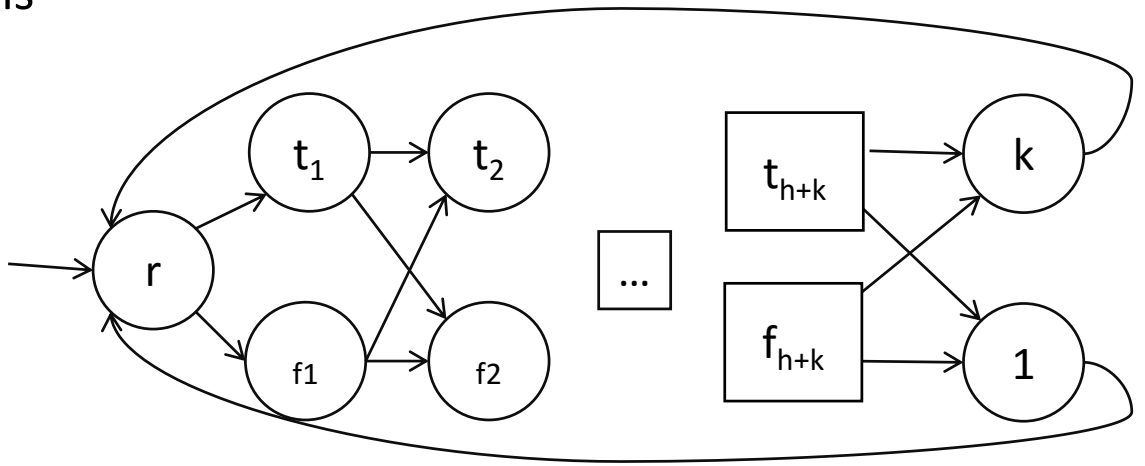


Complexity EXPTIME-easy

- Similar to BSIL, Let |x| be the length of the TCL formula
- The maximum depth of the computation tree will be $|Q|*|X|*2^{|X|}$

Complexity-TCL model checking is EXPTIME-hard

- Reduction from PEEK-G_6 game[1979]:
  - Propositions $p_1 \ldots p_{h+k}$
  - Safety agent control P1=$\{p_1, p_2 \ldots\}$
  - Reachability agent control P2=$\{p_{h+1}, p_{h+2} \ldots p_{h+k}\}$
  - The Reachability agent wants to satisfy a Boolean formula β over P1$\cup$P2
  - In each turn, each agent can change the truth value of one of his own propositions



Fault Tolerance

- More than checking whether or not a system satisfies a given property
  - Bounded number of failures:
    Check the max number of failures a system can tolerant before exhibit an error
  - Unbounded number of failures:
    Given a bound on the number of “dense failures”.
    Check if the system can be “fully recovered” after these dense failures

Error Models

- Fault tolerance refers to various basic fault models, such as a limited number of errors.
  [H. Jin, K. Ravi, F. Somenzi]
- Robustness based on Hamming and Lewenstein distance related to the number of past states.
  [L. Doyen, T.A. Henzinger, A. Legay, D. Nickovic]
- Ratio Games: minimize the ratio between failures induced by the environment and system errors caused by them. *[R. Bloem, K. Greimel, T.A. Henzinger, B. Jobstmann]*

Dense Fault

- Two successive failures are in the same *group of dense failures* if the sequence of states separating them was not long enough for recovery in the respective safety/reachability game.

Why dense fault

k	0	1	2	3	4	5	6	...
k errors	0.865	0.594	0.333	0.143	0.053	0.017	0.005	...
k dense errors	0.865	$2 \cdot 10^{-4}$	$2 \cdot 10^{-9}$	$2 \cdot 10^{-14}$	$2 \cdot 10^{-19}$	$2 \cdot 10^{-24}$	$2 \cdot 10^{-29}$	...

- System Operation time: 20 hour
- Mean time between errors: 10 hour
- Repair time: 3.6 sec
- Since errors occur with a known average rate and independently of the time since the last event, we use Poisson distribution(with coefficient 2) for "*k errors*" in the above table

Recovery Segment

- A play prefix ρ is a recovery segment to safety region $S \subseteq Q \setminus F$ iff
  - $\rho(0) \in S$
  - If $|\rho| = \infty$, then all states in $\rho[1,\infty)$ are in $Q\setminus(S \cup F)$.
    In this case, ρ is called a failed recovery segment.
  - If $|\rho| \neq \infty$, then all states in $\rho[1, |\rho| - 2]$ are in $Q \setminus (S \cup F)$ and $last(\rho) = \rho(|\rho| - 1)$ is either in F or S.
    If $last(\rho) \in F$, ρ is also a failed recovery segment;
    otherwise, it is a successful one.
- level(ρ, S): *the number of error moves between states in* ρ *with respect to the safety region* S*:*
  $\{i \in [0, |\rho| - 1) \mid \rho\_e(i) \models E\mathit{error}\}$

Gain

- gain(ρ, S): The maximal integer k ∈ N such that, for all recovery segments ρr to S in ρ, if level(ρr, S) ≤ k, then ρr is a successful recovery segment to S.
- Gain(*Γ*,S): the maximum gain that the protagonist can manage with memoryless strategies on game graph K.
- *Γ is* k-resilient *if there exists a non-empty* S ⊆ Q \ F *with* gain(*Γ*, S) ≥ k.

AMCE Alternating-time μ-calculus with events

- AMCE is an extension of AMC
- AMC example: μX(*safe* ∨ <1> ○ X)
- Extension 1: Boolean combination
  <1>((smoke ⇒ ○ alarmOn) ∨ ○ windowClosed)
- Extension 2: Restriction on transitions
  <1>((○ $^{2:error}$*alarmOn*) ∧ (○ $^{\neg 2:error}$$^{\neg}$*alarmOn*))

Algorithm-Base case

- Safety and Reachability Objective, $sfrch_k(S)$, denotes the states from which the protagonist wins the above game
- A state $q \in S$ can stay in $sfrch_0(S)$ if there is a choice $e \in E1$ such that for all $f \in E2$, $\delta(q, e, f) \in sfrch_0(S)$.
- $sfrch_0(S) = \nu X(S \wedge \langle 1 \rangle \bigcirc^{error} x)$
- $sfrch_0(S)$ can be constructed by greatest fixed point algorithm

Algorithm - Inductive Case - cone

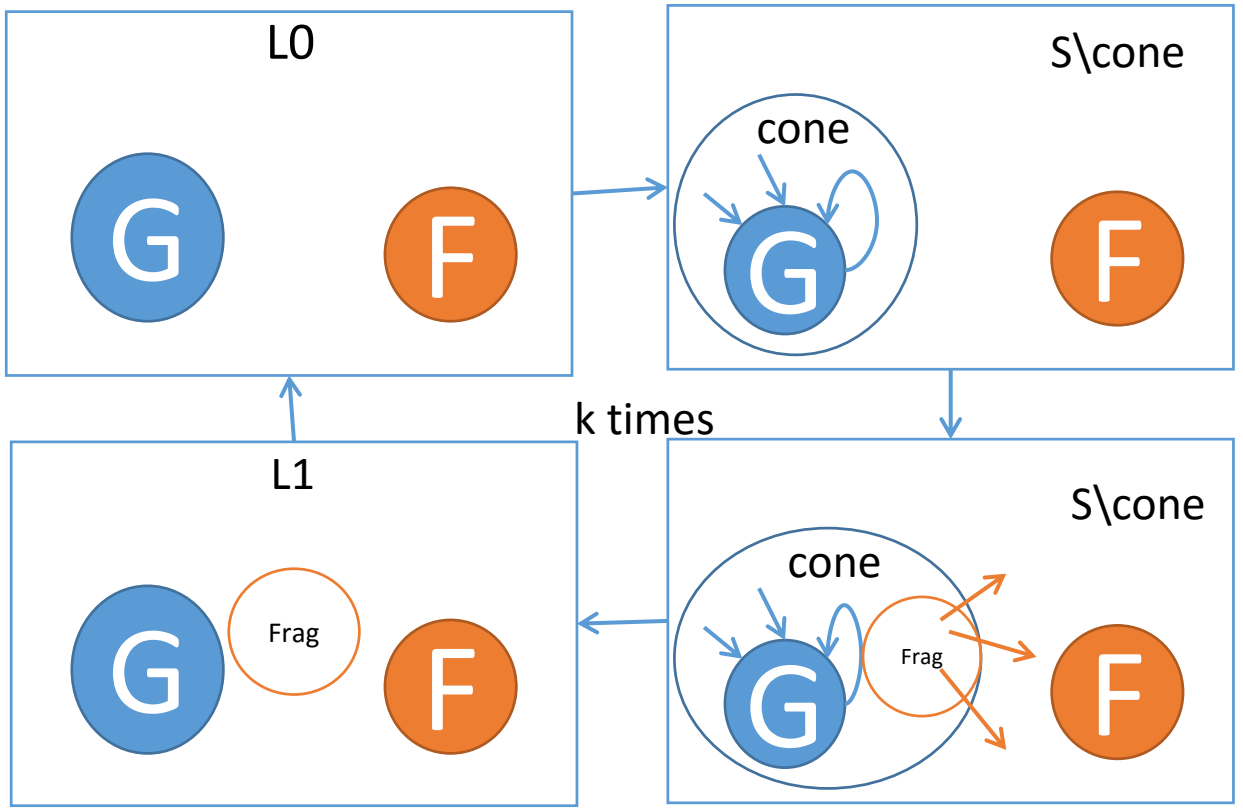
- Given a region L⊑S\F, the controlled limited attractor,
  is a set of states which there is a controlled path to move to X without leaving L.
- $cone_L(X) = \mu Y.(X \vee (L \wedge \langle 1 \rangle \bigcirc^{\neg 2:error} Y))$
- $cone_L(X)$ can be constructed by greatest fixed point

Algorithm - Inductive Case - frag

- Given a set B⊆S, the fragile of B, frag(B), is the set of states which has more than 1 uncontrolled successor in B.
- frag(B) = [1] ○ ¬B
- Q \ frag(B) = <1> ○ B.
- frag(B) and Q\frag(B) can be constructed with least fixed algorithm

Algorithm - Inductive Case - $sfrch_k(S)$ and $res_k(S)$

- $L_k = L_0 \setminus frag(Q \setminus cone_{Lk-1}(S))$
- $sfrchk(S) = sfrch0(S \wedge L_k)$



- $resk(G) = \nu S.((Q \setminus F) \wedge sfrchk(S))$, the set of k-resilient states

Complexity

- kmax *is either infinite or no greater than* |Q\F|*.*
  - *if k>* |Q\F| *-> exists fail recovery path* ρ *with k+1 states*
  - *However, there are only* |Q\F| *safety states -> repeat states in the path*
  - There should be a shorter fail path -> contradiction
- *A memoryless control strategy for the states in* sfrchk(S) *can be constructed in time linear in both* k *and the Game size* |G|
  - All individual steps in the construction are linear in the size of the safety resilience game, and there are O(k) of these operations in the construction.
- resk(G) *and a memoryless* k*-resilient control strategy for* resk(G) *can be constructed in* O(k·|Q\F|·|G|) *time.*
  - There are at most |Q\F| times of sfrchk(S) during the process of finding resk(G)

Experiment-BSIL

Properties	Parameters									Result	Time	Mem
	s	c_1	b_1	c_2	b_2	c_3	b_3	c_4	b_4			
(L)	1	2	2	2	4	0	0	0	0	UNSAT	0.52s	61M
	2	2	4	2	4	0	0	0	0	UNSAT	1.20s	75M
	3	2	4	2	4	0	0	0	0	SAT	0.73s	75M
	3	3	6	3	6	0	0	0	0	SAT	1.51s	211M
	3	3	9	3	6	0	0	0	0	UNSAT	9.14s	837M
	3	3	6	3	9	0	0	0	0	SAT	8.74s	502M
	4	3	9	3	9	0	0	0	0	SAT	158s	6631M
(M)	1	2	2	2	4	0	0	0	0	UNSAT	0.53s	61M
	2	2	4	2	4	0	0	0	0	UNSAT	1.15s	75M
	3	2	4	2	4	0	0	0	0	UNSAT	1.14s	75M
	3	3	6	3	6	0	0	0	0	UNSAT	0.93s	211M
	3	3	9	3	6	0	0	0	0	SAT	5.22s	681M
	3	3	6	3	9	0	0	0	0	UNSAT	8.49s	502M
	4	3	9	3	9	0	0	0	0	UNSAT	93.17s	5082M
(N)	2	2	6	2	3	2	2	0	0	UNSAT	2.86s	493M
(O)										SAT	4.61s	368M
(P)	3	2	6	2	3	2	3	2	2	UNSAT	179s	3755M
(Q)										SAT	209s	2862M
(R)										SAT	75s	1329M

Experiment-TCL

properties \ m	2	3	4	5	6	7	8	9	10
(A)	0.71s	0.94s	5.41s	66.3s	945s	>1000s			
	163M	165M	185M	350M	1307M				
(B)	0.50s	0.52s	0.61s	0.71s	1.11s	1.62s	5.77s	20.9s	68.1s
	163M	163M	164M	165M	168M	176M	214M	270M	376M
(C)	0.51s	0.51s	0.6s	0.82s	1.01s	1.81s	5.54s	18.2s	48.3s
	163M	163M	164M	165M	168M	176M	200M	241M	318M
(D)	0.5s	0.51s	0.57s	0.74s	1.01s	1.79s	7.41s	33.8s	141s
	163M	163M	164M	165M	168M	175M	232M	312M	430M
(E)	0.51s	0.66s	19.1s	>1000s					
	163M	164M	194M						
(F)	0.51s	0.53s	0.61s	0.71s	1.01s	1.70s	5.38s	15.2s	53.7s
	163M	163M	163M	165M	168M	175M	202M	243M	295M
(G)	0.52s	0.52s	0.65s	0.72s	1.03s	1.85s	4.86s	16.1s	93.5s
	163M	163M	164M	165M	169M	177M	189M	208M	235M

s: seconds; M: megabytes.

Experiment-Resilience

benchmarks	concurrency	k	game sizes		sfrch$_k$		res$_k$	
			#nodes	#edges	time	memory	time	memory
avionics	2 processors & 2 memory modules	2	118	750	0.62s	114M	0.85s	116M
	2 processors & 3 memory modules	2	414	3252	0.94s	139M	1.10s	153M
	3 processors & 3 memory modules	3	1540	15090	4.67s	225M	8.38s	267M
	3 processors & 4 memory mdules	3	5601	63889	42.86s	815M	155s	846M
avionics (counter abstraction)	6 processors & 6 memory modules	2	1372	6594	2.89s	129M	3.54s	516M
	7 processors & 7 memory modules	3	2304	11396	10.7s	216M	23.4s	808M
	8 processors & 8 memory modules	3	3645	18432	43.8s	1009M	135s	2430M
voting	1 client & 20 replicas	9	9922	23551	7.01s	260M	36.7s	297M
	1 client & 26 replicas	12	20776	49882	19.9s	474M	79.6s	611M
simple voting	1 client & 150 replicas	74	458	1056	0.71s	159M	31.7s	219M
	1 client & 200 replicas	99	608	1406	1.06s	161M	162s	337M
	1 client & 250 replicas	124	758	1756	1.36s	163M	307s	499M
PBFT	1 client & 6 replicas	2	577	897	0.34s	72M	1.05s	193M
	1 client & 9 replicas	4	2817	4609	13.3s	564M	58.5s	1657M
clock sync	1 client & 15 servers	7	16384	229376	45.1s	3075M	62.4s	3264M
	1 client & 17 severs	8	65536	1070421	870s	14725M	915s	15433M

Conclusion

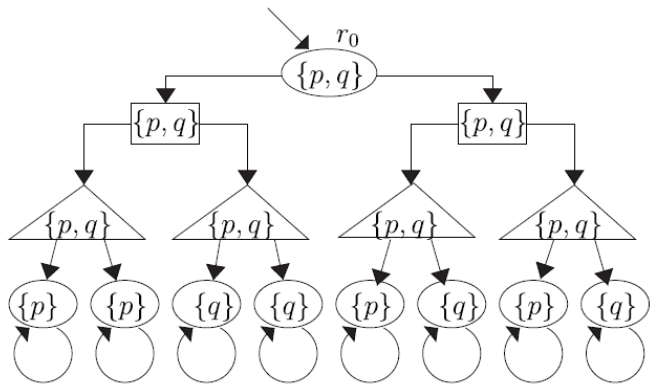
- BSIL and TCL reach a balance between expressiveness and verification efficiency with the capability to describe strategy inherit/release properties
- The dense error resilience defines a new error model which and can be verified in PTIME.

Q&A

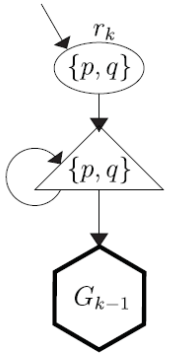
Backup

Expressive power

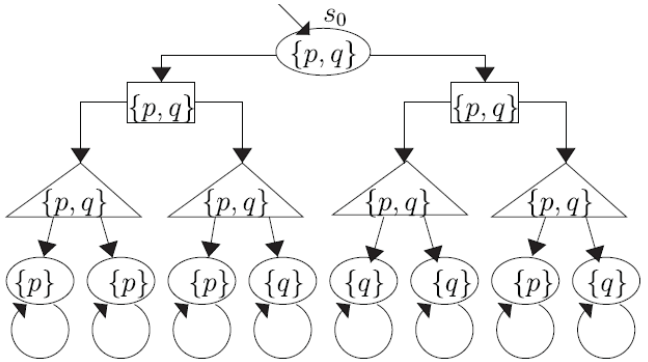
- BSIL: <1>((<+2>□*p*)∧(<+2>□*q*))
- AMC



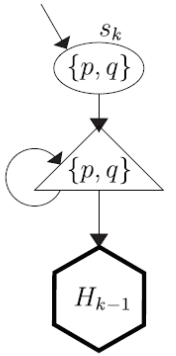
(a) G_1, a game graph for base case.



(a) G_k



(b) H_1, another game graph for base case.



(b) H_k

◯ belongs to Agent 1; □ belongs to Agent 2; and △ belongs to Agent 3.

Temporal Cooperation Logic(TCL)

- Allow strategy interaction quantifiers to cross temporal modal operators
- $\langle 1,2,3\rangle \bigwedge_{a\in[1,3]} ((\langle +\varnothing\rangle \Diamond \neg \text{jail}_a) \vee (\langle -a\rangle \Box \text{jail}_a))$
  no agent stays in jail indefinitely, if she can avoid it. (Nash equilibrium)
- $\langle 2\rangle((\langle +\rangle \Box \neg \text{betray}_2)$
  $\vee \langle +1\rangle \Diamond((\langle +\rangle \bigcirc \neg \text{betray}_1) \wedge (\text{betray}_2 \vee \text{betray}_3)))$
  Player 2 should avoid betrayal while player 1 can be unforgiving