Indiana Jones and . . .

Defining Redundancy

In Search of the Holy Grail

An attempt to understand the Arora/Kulkarni fault-tolerance theory

Felix Gärtner

TU Darmstadt, Germany

(most of this work done with Hagen Völzer, HU Berlin)

Indiana Jones and . . .

Defining Redundancy

In Search of the Holy Grail

An attempt to understand the Arora/Kulkarni fault-tolerance theory

Felix Gärtner

TU Darmstadt, Germany

(most of this work done with Hagen Völzer, HU Berlin)

Danger: Contains unproved theorems, but

Indiana Jones and . . .

Defining Redundancy

In Search of the Holy Grail

An attempt to understand the Arora/Kulkarni fault-tolerance theory

Felix Gärtner

TU Darmstadt, Germany

(most of this work done with Hagen Völzer, HU Berlin)

Danger: Contains unproved theorems, but

100% pure Dagstuhl

Motivation

- Fault-tolerance is a complex field with lots of complicated mechanisms.
- Difficulty of teaching fault-tolerance to students or attracting researchers.
- 1998 work of Arora and Kulkarni [2]: theory of detectors and correctors.
- Nice framework to describe how things work in fault tolerance.
- Difficult to understand intricacies.

Motivation

- Fault-tolerance is a complex field with lots of complicated mechanisms.
- Difficulty of teaching fault-tolerance to students or attracting researchers.
- 1998 work of Arora and Kulkarni [2]: theory of detectors and correctors.
- Nice framework to describe how things work in fault tolerance.
- Difficult to understand intricacies.
- Offers nice explanation of the concept of *redundancy*.

Overview

- Preliminaries (states, traces, properties, programs, etc.) (5 slides)
- Fault models and fault-tolerant versions (4 slides)
- Safety, detectors and redundancy in space (4 slides)
- Liveness, correctors and redundancy in time (4 slides)
- Conclusions (1 slide)

States and Traces

top

- State set C (countable)
- State predicate φ over C: subset of C
- State transition over C: $(s, s') \in C \times C$
- Trace over C: non-empty infinite sequence $\sigma = s_0, s_1, s_2, \ldots$ of states from C

Properties

- Property over C: set of traces over C
- Safety property S:

$$\sigma \notin S \Rightarrow \exists \text{ prefix } \alpha \text{ of } \sigma \text{ s.t. } \forall \beta \text{ holds } \alpha \cdot \beta \notin S$$

- Liveness property L:

$$\forall \text{ finite traces } \alpha\ \exists \beta \text{ s.t. } \alpha \cdot \beta \in L$$

- Every property is the intersection of a safety property and a liveness property [1].

Programs and Liveness Assumptions

- Program $\Sigma = (C, I, T, A)$: state set C, initial states $I \subseteq C$, transitions $T \subseteq C \times C$, liveness assumption A

- Liveness assumption A for Σ: liveness property over C such that

  $\forall$ finite traces α of Σ $\exists\beta$ s.t. $\alpha \cdot \beta \in A$ and $\alpha \cdot \beta$ is a trace of Σ

- Fairness assumptions are special forms of liveness assumptions.

- Extend finite traces to infinite traces by infinitely repeating final state.

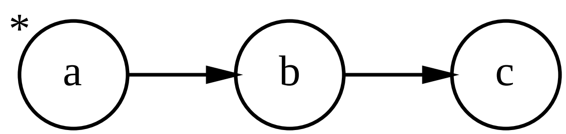
- (C, I, T) define a safety property S. Property of Σ: $prop(\Sigma) = S \cap A$

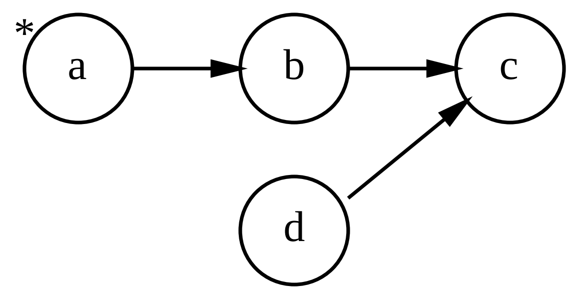
Specifications and Correctness

- Set X is fusion closed iff $\alpha \cdot s \cdot \beta \in X$ and $\gamma \cdot s \cdot \delta \in X$ implies $\alpha \cdot s \cdot \delta \in X$ and $\gamma \cdot s \cdot \beta \in X$
- Specification $SPEC$: fusion closed property
- Specifications can be made fusion closed using history variables.
- Σ satisfies $SPEC$: $prop(\Sigma) \subseteq SPEC$
- Σ violates $SPEC$: Σ not satisfies $SPEC$

Extensions of Programs

- Program $\Sigma_2 = (C_2, I_2, T_2, A_2)$ extends $\Sigma_1 = (C_1, I_1, T_1, A_1)$ iff
  - $C_2 \supseteq C_1$
  - $A_2 = A_1$
  - $prop(\Sigma_1) = prop(\Sigma_2)$

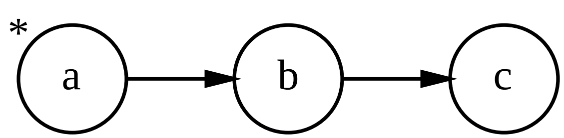




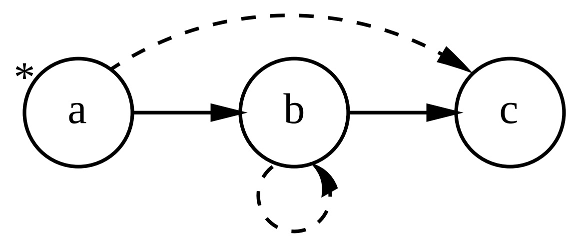
Fault Models

top

- $\mathcal{T}$ = set of all transition systems
- Fault model: function $F : \mathcal{T} \rightarrow \mathcal{T}$
- $F((C, I, T, A)) = (C, I, T', A')$ with:
  - $T \subseteq T'$
  - $A \subseteq A'$



Σ



$F(\Sigma)$

Completeness of Fault Model

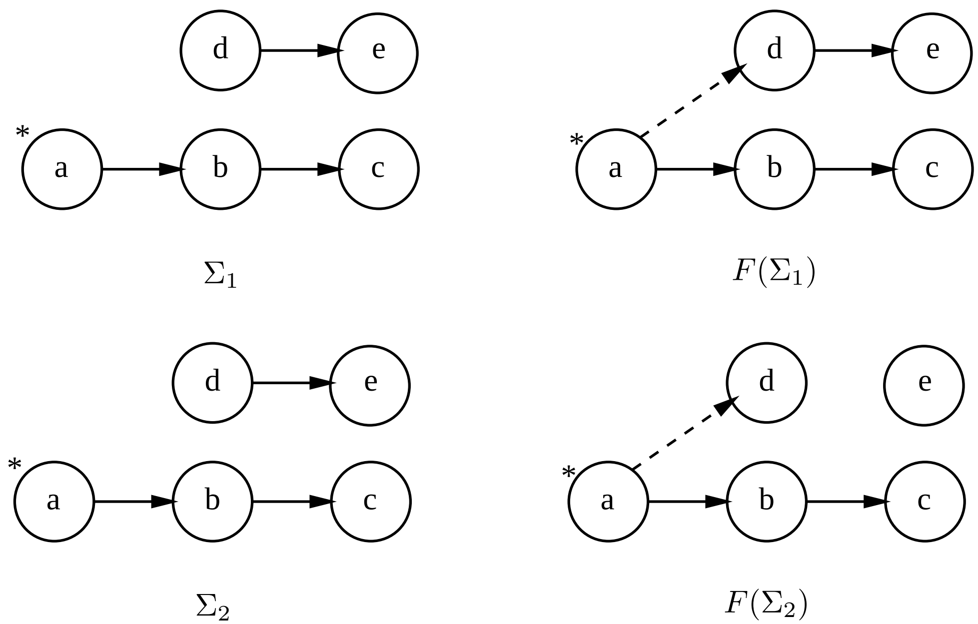
- Completeness theorem (unproved): "all" faulty behaviors can be implemented by some F.

  Theorem. *Given a transition system* Σ*, faulty behavior* $P \supset prop(\Sigma)$ *with no new initial states, then* $\exists F$ *s.t.* $prop(F(\Sigma)) = P$*.*

- Proof idea: violate safety by adding transitions, violate liveness by (adding transitions and) weakening liveness assumption.

Fault-tolerant Versions

- Σ_2 is an F-tolerant version of Σ_1 for $SPEC$ iff
  - Σ_2 extends Σ_1
  - Σ_1 satisfies $SPEC$
  - $F(\Sigma_1)$ violates $SPEC$
  - $F(\Sigma_2)$ satisfies $SPEC$



Fusion-Closure and Safety

top

- Fusion closure gives you a set of "bad transitions":

  Lemma. *If*

  – $SSPEC$ *is a* fusion-closed *safety specification,*
  – $\Sigma = (C, I, T, A)$ *violates* $SSPEC$*,*
  – *all initial states of* Σ *maintain* $SSPEC$

  then $\exists$ *transitions* $t \in T$ *s.t.* $\forall$ *traces* σ *of* Σ *holds* t *occurs in* $\sigma \Rightarrow \sigma \notin SSPEC$

- Bad transitions (s, s') can be avoided iff s is a non-reachable program state.

Fault Models and Safety

skip

Lemma. *Take some fault model F. If*

- $SSPEC$ *is a safety specification,*
- $\Sigma = (C, I, T, A)$ *satisfies* $SSPEC$*,*
- $F(\Sigma)$ *violates* $SSPEC$

then F *adds at least one transition to* T*.*

- No need of fusion closure.
- Adding transitions sufficient to violate safety.

Introducing Detectors

top

- A *detector* is a program module which detects whether a predicate is true on the system state.

- Detectors can be composed from smaller detectors.

Theorem. *[3, p. 28] Detectors are sufficient for satisfying safety specifications.*

Theorem. *[3, p. 33] Fault-tolerant versions contain detectors.*

Explanation of Detectors

Theorem. *If* Σ_2 *is an* F*-tolerant version of* Σ_1 *for a safety specification* $SSPEC$
then C_2 *contains non-reachable states.*

- Detectors "cut away" F-reachable bad transitions.
- Notion of *state space redundancy*.

Definition. *A program employs redundancy in space iff it contains non-reachable states.*

Corollary. *Redundancy in space necessary for safety (or: detectors contain redundancy in space).*

Fault Models and Liveness

top

Lemma. *(unproved) Take some fault model F. If*

- $LSPEC$ *is a liveness specification,*
- $\Sigma = (C, I, T, A)$ *satisfies* $LSPEC$*,*
- $F(\Sigma)$ *violates* $LSPEC$

then F *(1) adds a transition to* T *or (2) adds traces to* A*.*

- Example for (2): Violation of liveness can be caused by assuming weak fairness instead of strong fairness.

Restricting Liveness

Lemma. *If*

- $LSPEC$ *is a liveness specification of the form* $\Diamond\Box\varphi$ *and*
- $\Sigma = (C, I, T, A)$ *violates* $LSPEC$

then $\exists$ *trace* σ *and a transition* $t = (s, s') \in T$ *such that* $\varphi(s')$ *holds and* t *occurs infinitely often*

- Infinitely often leave φ-states.
- What about other forms of liveness?

Introducing Correctors

- A *corrector* is a program module which "brings" the system into a certain state.

- Correctors can be composed from smaller correctors.

Theorem. *[3, p. 47] Correctors are sufficient for eventual satisfaction of a specification.*

Theorem. *[3, p. 51] Fault-tolerant versions for liveness specifications of the form $\Diamond\Box\varphi$ contain correctors.*

Explanation of Correctors

Theorem. *(unproved) If* Σ_2 *is an* F*-tolerant version of* Σ_1 *for liveness specification* $LSPEC$ *of the form* $\Diamond\Box\varphi$ *then* C_2 *contains non-reachable states and* T_2 *contains non-reachable transitions.*

- Correctors add transitions that "go to" φ-states.
- Notion of *time (or transition) redundancy*.

Definition. *A program employs redundancy in time iff it contains non-reachable transitions.*

Corollary. *Redundancy in time necessary for satisfying* $\Diamond\Box\varphi$ *(or: correctors contain redundancy in time).*

Conclusions

top

- Detector/corrector theory is complex.
- Offers nice framework to explain how things work in fault-tolerance.
- Offers possibility to formally define space and time redundancy.
- Future work:
  - Prove the theorems (???).
  - Introduce measures of "redundancy-ness" to compare protocols (???).

Conclusions

top

- Detector/corrector theory is complex.
- Offers nice framework to explain how things work in fault-tolerance.
- Offers possibility to formally define space and time redundancy.
- Future work:
  - Prove the theorems (???).
  - Introduce measures of “redundancy-ness” to compare protocols (???).
- . . . search for the holy grail continues. . .

References

[1] Bowen Alpern and Fred B. Schneider. Defining liveness. *Information Processing Letters*, 21:181–185, 1985.

[2] Anish Arora and Sandeep S. Kulkarni. Component based design of multitolerant systems. *IEEE Transactions on Software Engineering*, 24(1):63–78, January 1998.

[3] Sandeep S. Kulkarni. *Component Based Design of Fault-Tolerance*. PhD thesis, Department of Computer and Information Science, The Ohio State University, 1999.

Acknowledgements

• Slides produced using "cutting edge" LaTeX slide processor PPower4 by Klaus Guntermann.