Second Generation Model-based Testing

Provably Strong Testing Methods for the Certification of Autonomous Systems
Part III of III –
Complete Test Suites for CSP Refinement

Jan Peleska
University of Bremen and Verified Systems International GmbH
peleska@uni-bremen.de
2019-03-21

Universität Bremen

Verified

Finite Complete Test Suites for CSP Refinement Testing*

Jan Peleska, Wen-ling Huang, and Ana Cavalcanti
{peleska,huang}@uni-bremen.de
ana.cavalcanti@york.ac.uk

*The results presented here have been submitted to Science of Computer Programming

Completeness Result

Theorem. Let P be a non-terminating, divergence-free CSP process over alphabet Σ whose normalised transition graph $G(P)$ has p states. Define fault domain $\mathcal{D}$ as the set of all divergence-free CSP processes over alphabet Σ, whose transition graph has at most q states with $q \geq p$. Then the test suite

$$TS_F = \{U_F(j) \mid 0 \leq j < pq\}$$

is complete with respect to $\mathcal{F} = (P, \sqsubseteq_F, \mathcal{D})$.

Analogous theorem holds for trace refinement

Recall

- **Complete test suites**
  - are specified for a given conformance relation
  - accept every conforming implementation
  - reject every non-conforming implementation

Recall

- **Fault domain.** A collection of models that may or may not conform to a reference model
- **Finite complete black-box test suites**
  - are specified with respect to a fault domain
  - guarantee completeness provided that the true SUT behaviour is reflected by a member of the fault domain
  - provide a **conformance proof** with finitely many finite test cases

Motivation

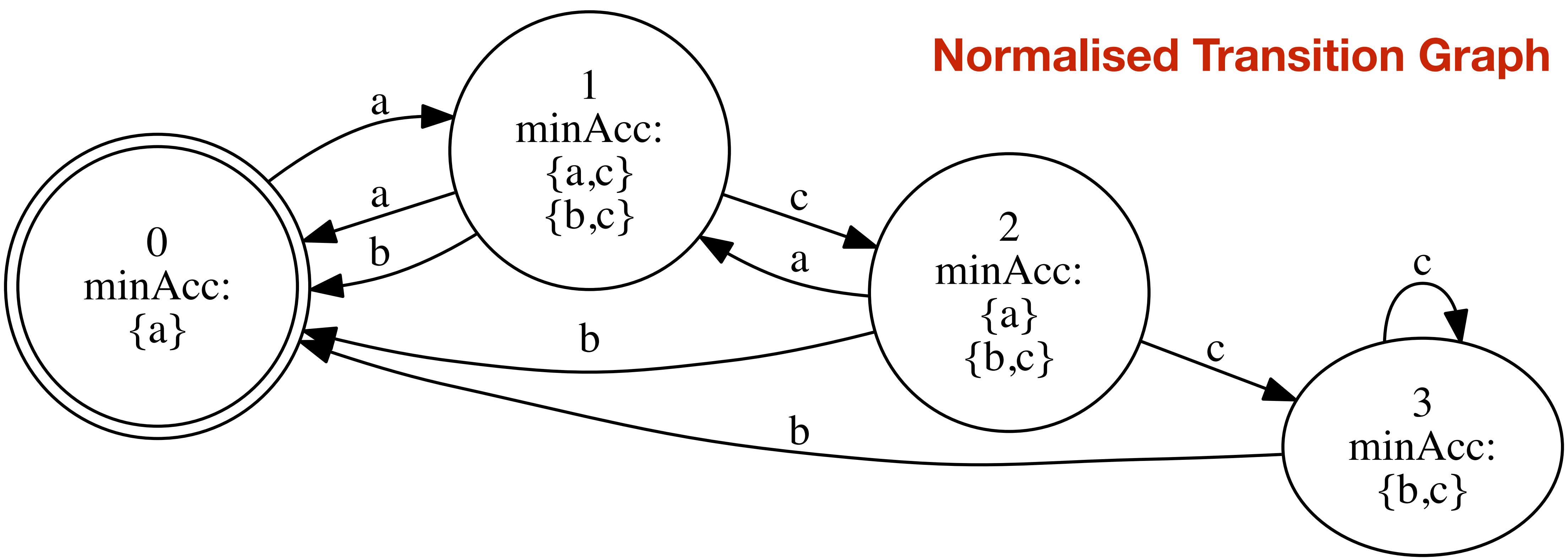
- Finite complete test suites are of high interest, because they
  - can **establish full conformance**, provided that the SUT behaviour is captured by the fault domain
  - still possess **high test strength for SUTs outside the fault domain** (experimental evidence)
  - still have manageable size if **equivalence class partitioning methods** are applied

Failures Semantics – Representation of Finite-State Processes

$$
\begin{aligned}
P &= a \to (Q \sqcap R) \\
Q &= a \to P \,\square\, c \to P \\
R &= b \to P \,\square\, c \to R
\end{aligned}
$$

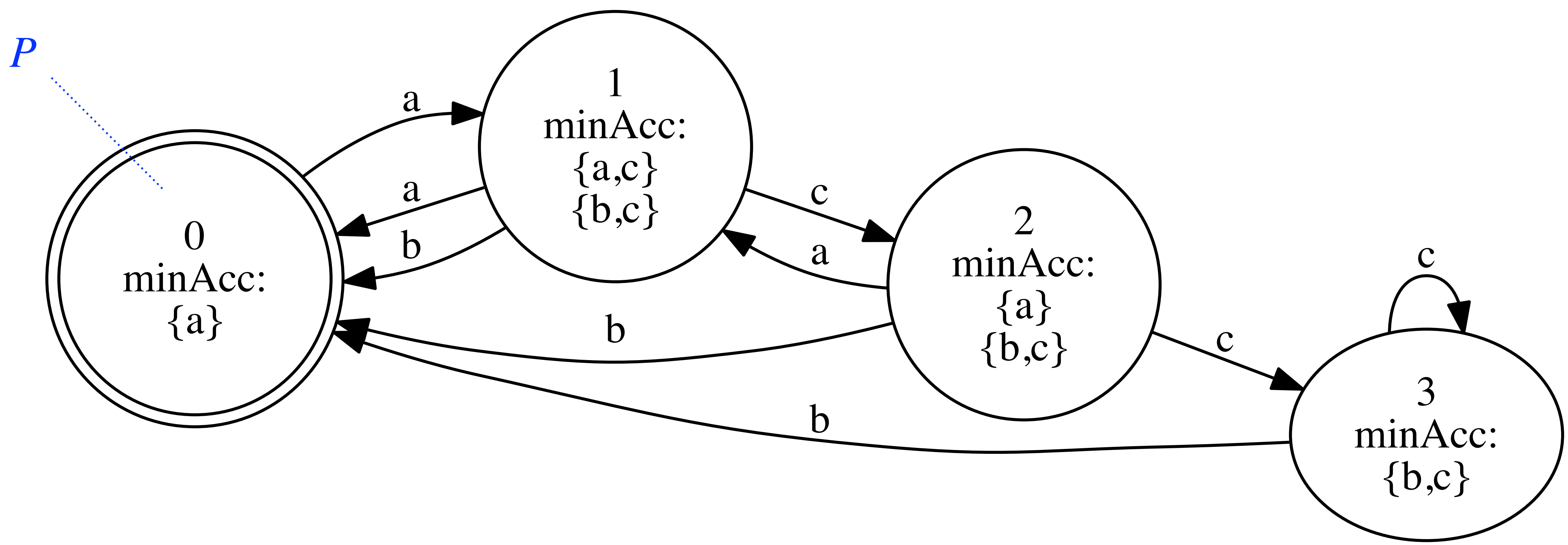
A. W. Roscoe, Model-checking CSP, in: A. W. Roscoe (Ed.), A Classical Mind: Essays in Honour of C. A. R. Hoare, Prentice Hall International (UK) Ltd., Hertfordshire, UK, UK, 1994, Ch. 21, pp. 353–378.

Normalised Transition Graph



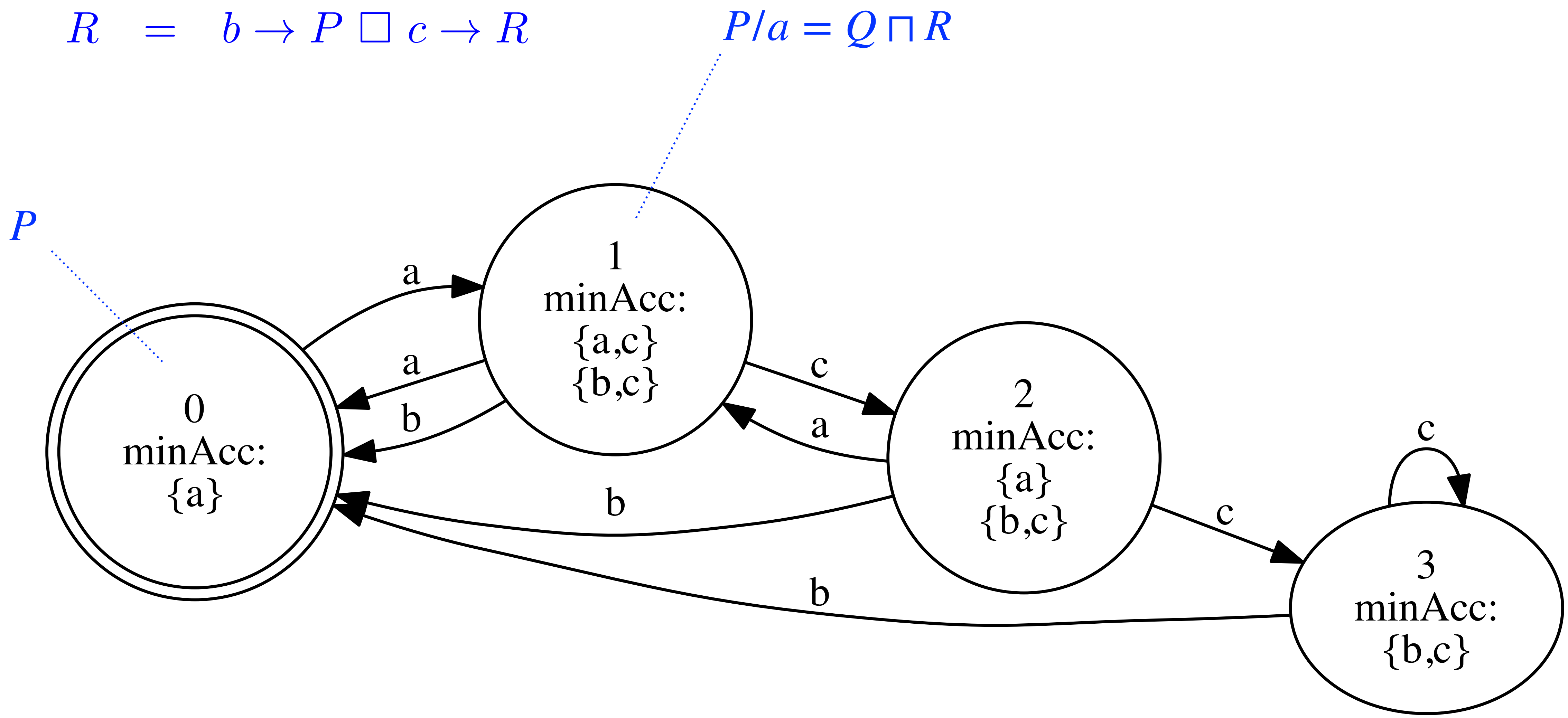
Failures Semantics – Representation of Finite-State Processes

$$
\begin{aligned}
P &= a \rightarrow (Q \sqcap R) \\
Q &= a \rightarrow P \;\Box\; c \rightarrow P \\
R &= b \rightarrow P \;\Box\; c \rightarrow R
\end{aligned}
$$



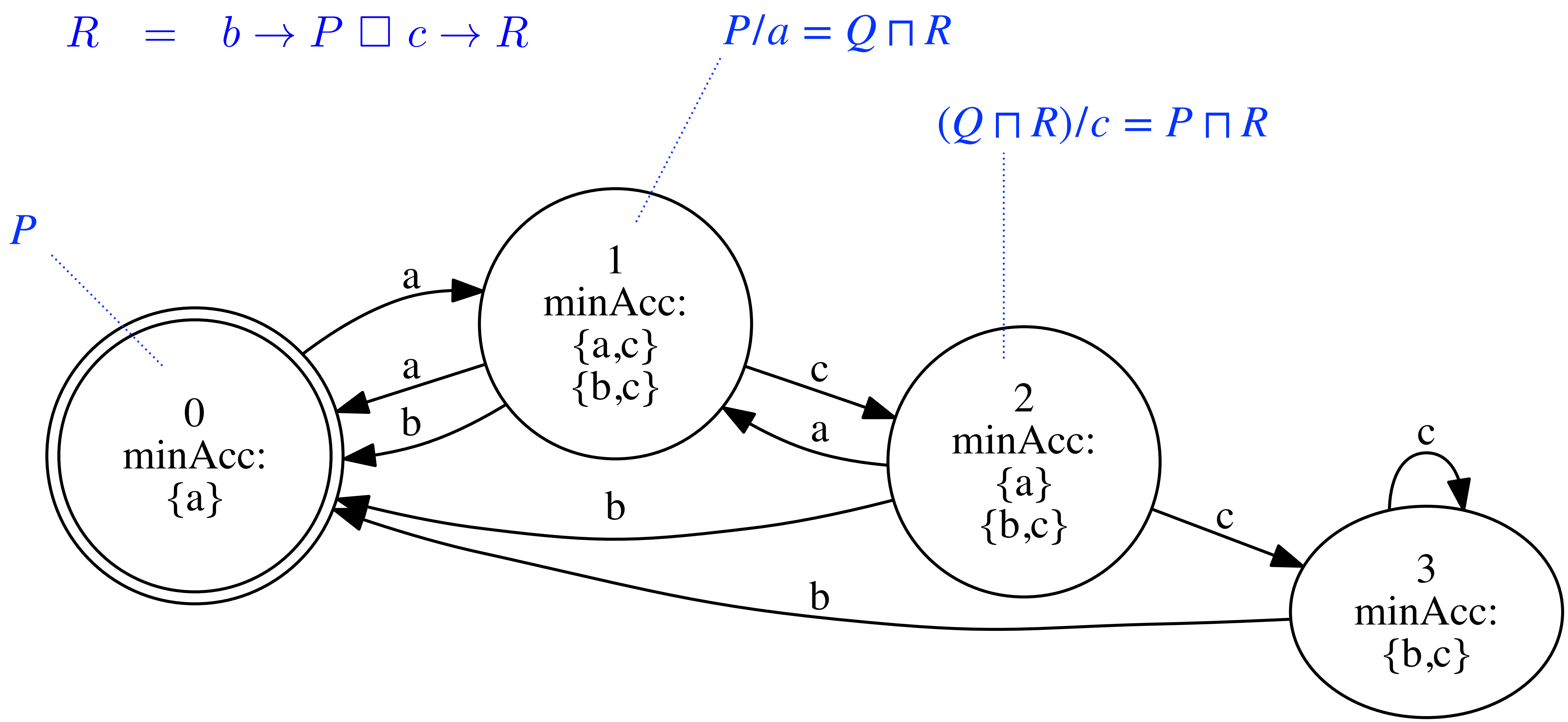
Failures Semantics – Representation of Finite-State Processes

$$
\begin{aligned}
P &= a \to (Q \sqcap R) \\
Q &= a \to P \,\Box\, c \to P \\
R &= b \to P \,\Box\, c \to R
\end{aligned}
$$



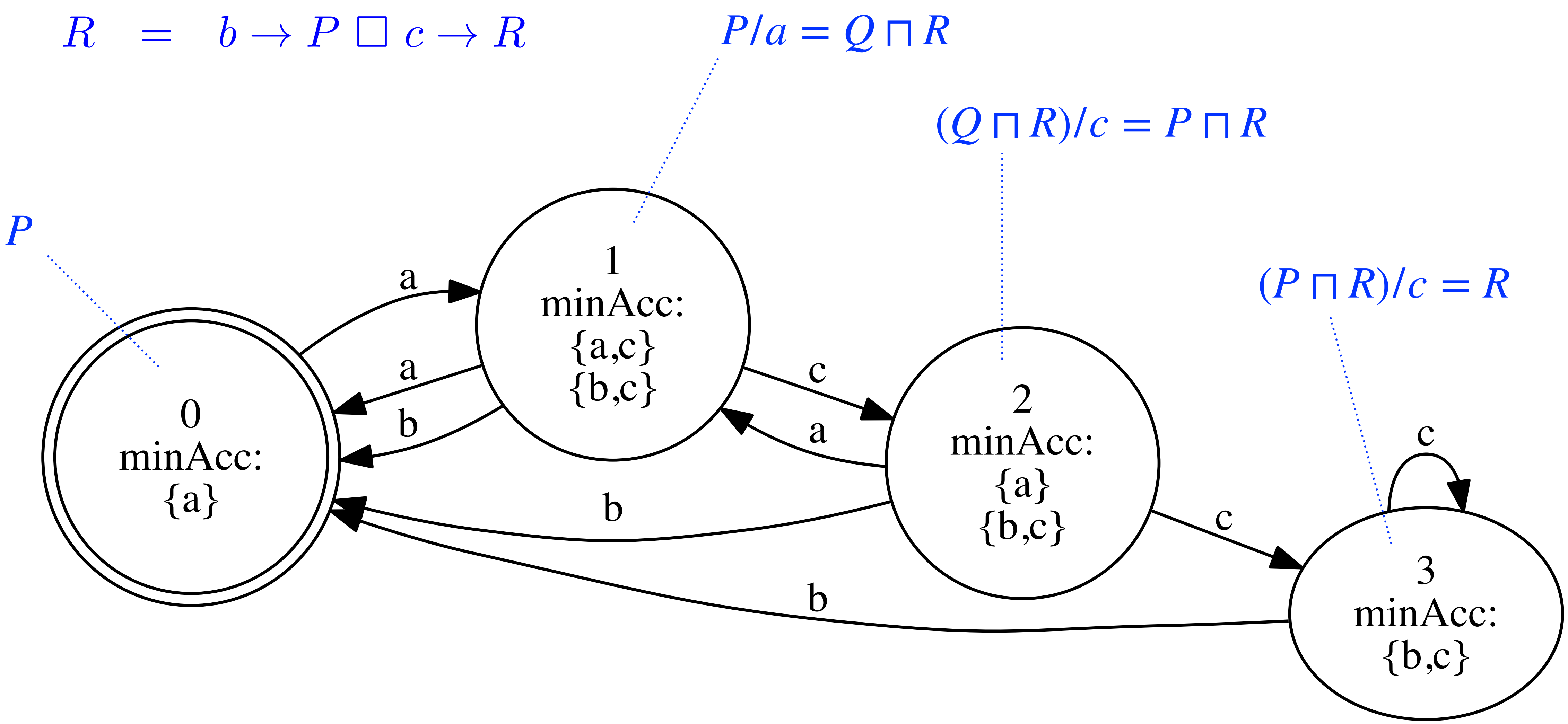
Failures Semantics – Representation of Finite-State Processes

$$
\begin{aligned}
P &= a \to (Q \sqcap R) \\
Q &= a \to P \;\Box\; c \to P \\
R &= b \to P \;\Box\; c \to R
\end{aligned}
$$



Failures Semantics – Representation of Finite-State Processes

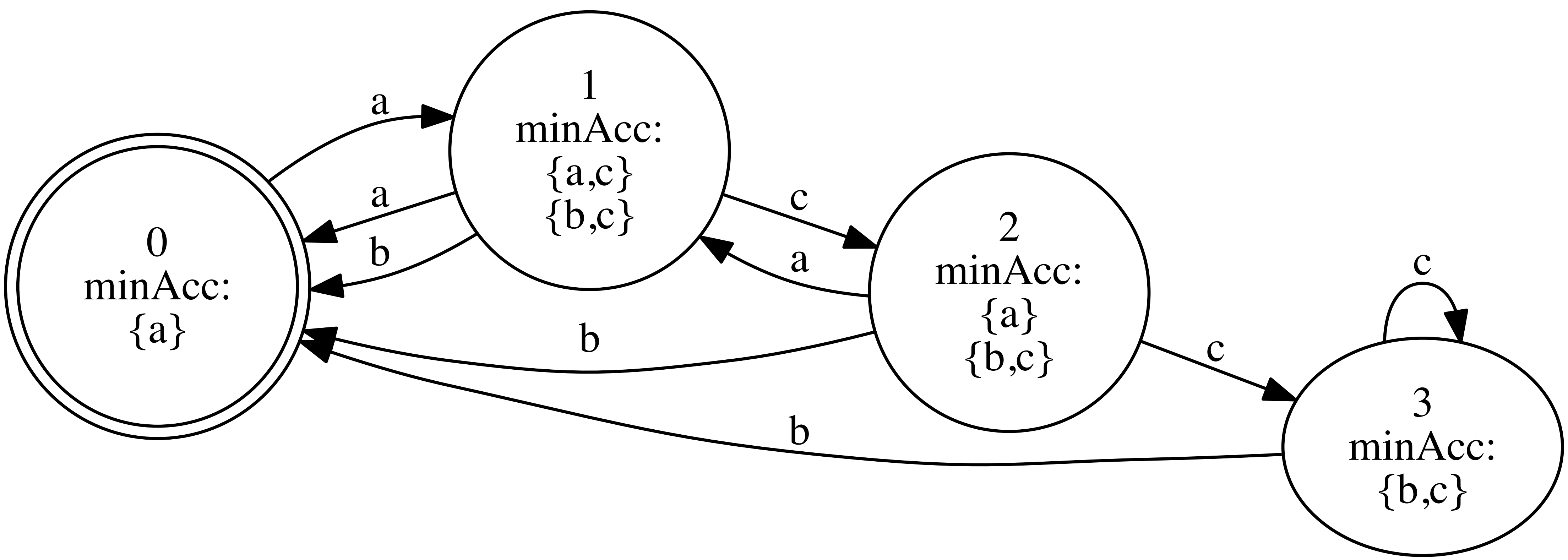
$$
\begin{aligned}
P &= a \rightarrow (Q \sqcap R) \\
Q &= a \rightarrow P \,\square\, c \rightarrow P \\
R &= b \rightarrow P \,\square\, c \rightarrow R
\end{aligned}
$$



Failures Semantics – Representation of Finite-State Processes

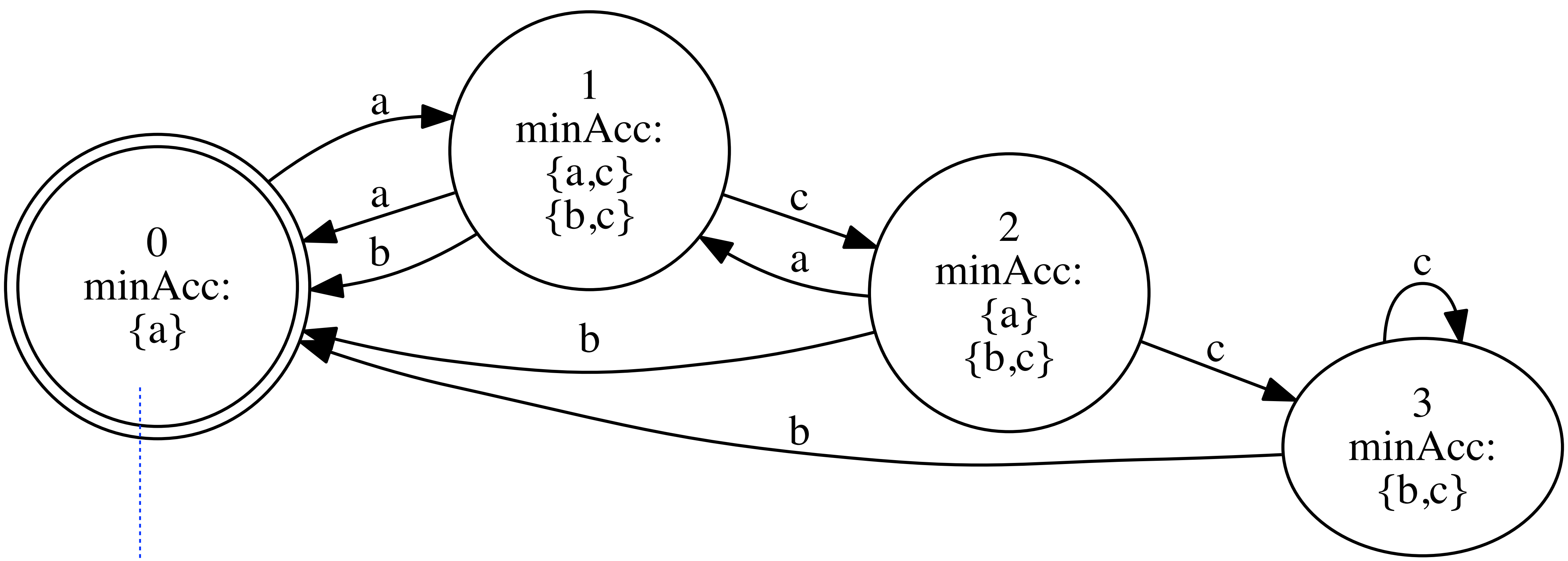
$$
\begin{aligned}
P &= a \rightarrow (Q \sqcap R) \\
Q &= a \rightarrow P \,\Box\, c \rightarrow P \\
R &= b \rightarrow P \,\Box\, c \rightarrow R
\end{aligned}
$$

Assume that implementation process Z has transition graph with 4 states – just like P



Failures Semantics – Representation of Finite-State Processes

$$
\begin{aligned}
P &= a \to (Q \sqcap R) \\
Q &= a \to P \,\Box\, c \to P \\
R &= b \to P \,\Box\, c \to R
\end{aligned}
$$

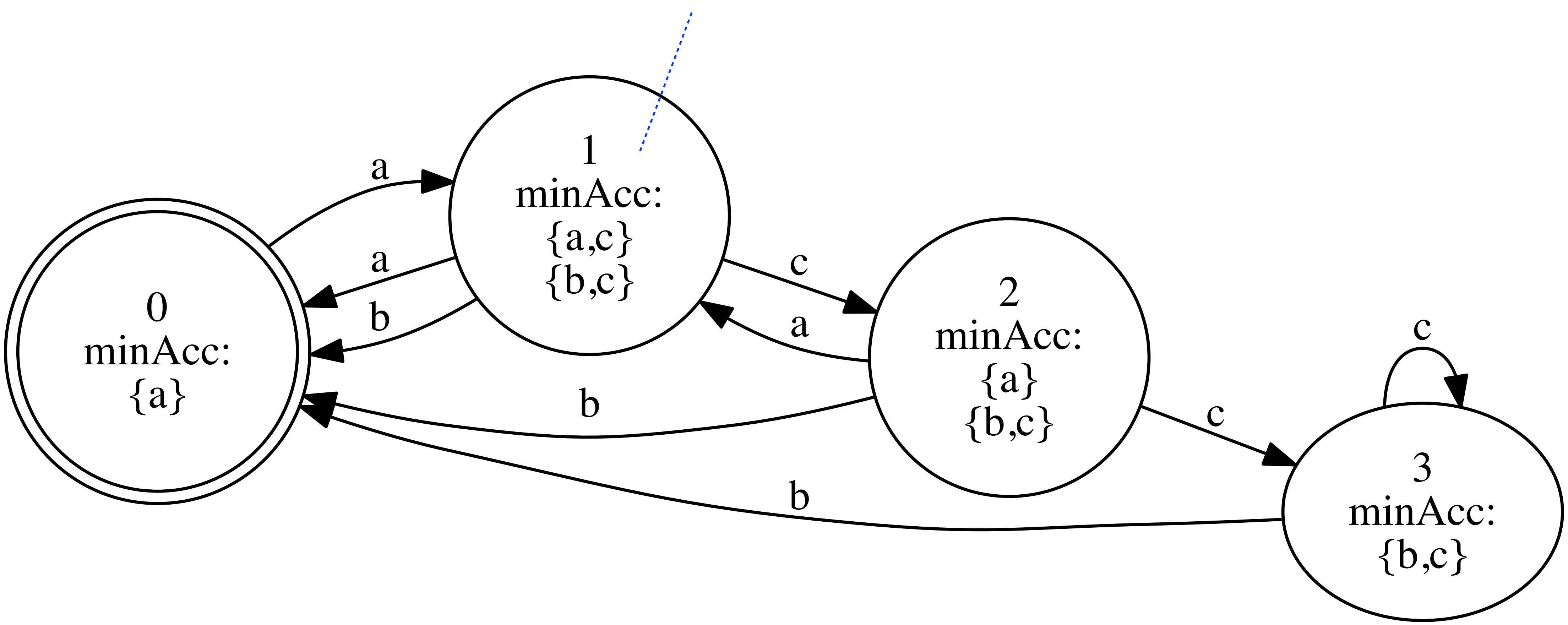


$\{a\}$ **must be a hitting set of** $acc(Z)$

Failures Semantics – Representation of Finite-State Processes

$$
\begin{aligned}
P &= a \rightarrow (Q \sqcap R) \\
Q &= a \rightarrow P \;\Box\; c \rightarrow P \\
R &= b \rightarrow P \;\Box\; c \rightarrow R
\end{aligned}
$$

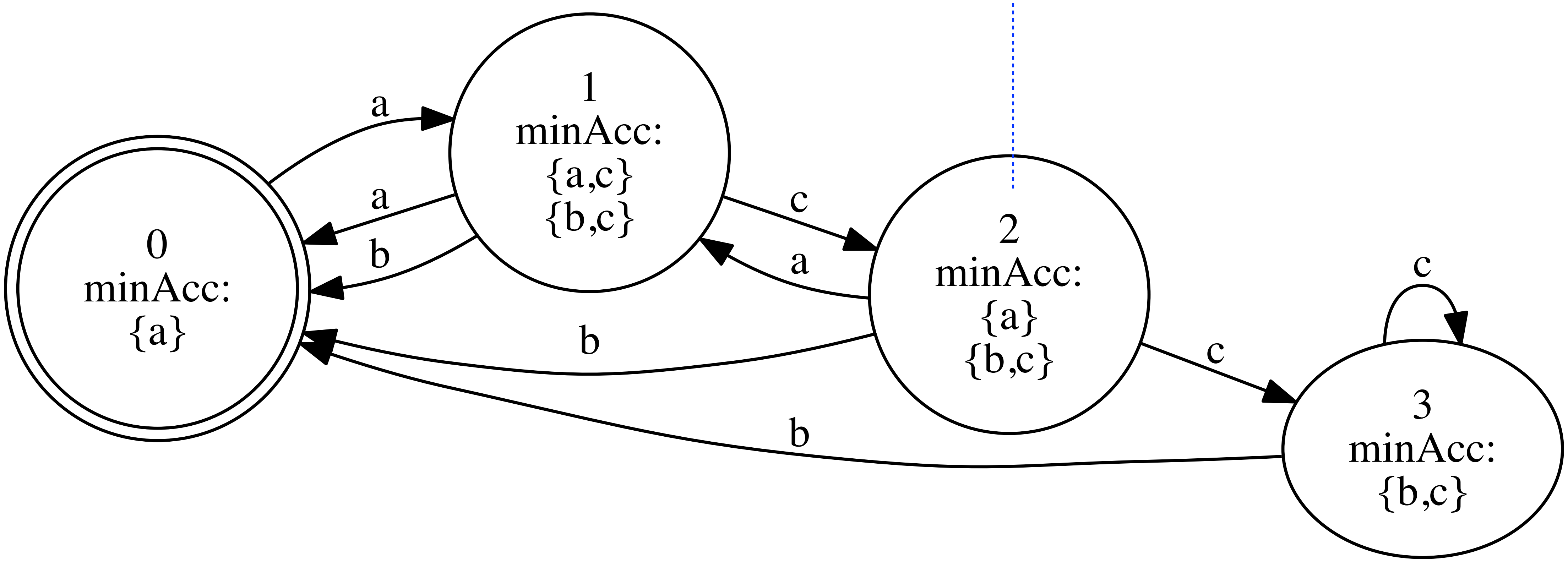
$\{a, b\}, \{c\}$ **must be hitting sets of** $acc(Z/a)$



Failures Semantics – Representation of Finite-State Processes

$$
\begin{aligned}
P &= a \rightarrow (Q \sqcap R) \\
Q &= a \rightarrow P \;\Box\; c \rightarrow P \\
R &= b \rightarrow P \;\Box\; c \rightarrow R
\end{aligned}
$$

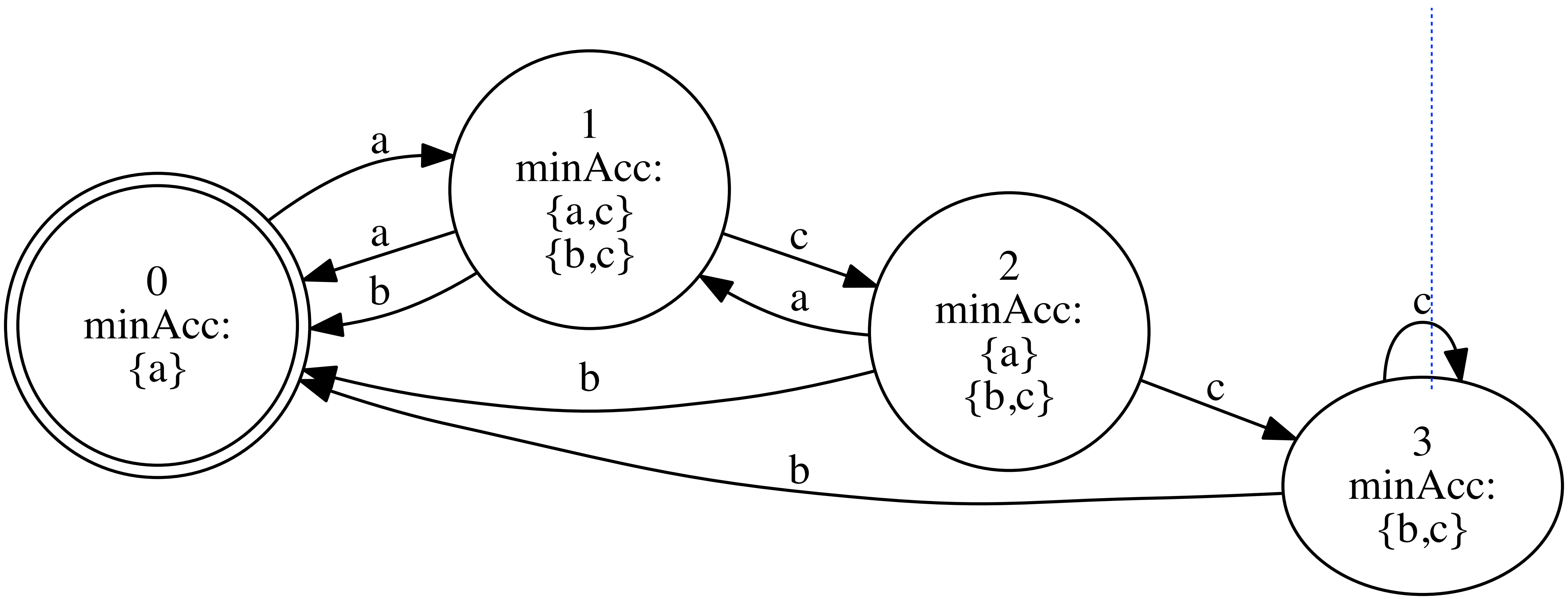
$\{a, b\}, \{a, c\}$ **must be hitting sets of** $acc(Z/a\,.\,c)$



Failures Semantics – Representation of Finite-State Processes

$$
\begin{aligned}
P &= a \to (Q \sqcap R) \\
Q &= a \to P \;\Box\; c \to P \\
R &= b \to P \;\Box\; c \to R
\end{aligned}
$$

$\{b\}, \{c\}$ **must be hitting sets of** $acc(Z/a\,.\,c\,.\,c)$



Adaptive Test Cases for Checking Failures Refinement

$$
\begin{array}{rcl}
U_F(j) & = & U_F(j, 0, \underline{n}) \\
U_F(j,k,n) & = & \big(e : (\Sigma - [n]^0) \to fail \to \mathbf{STOP}\big) \\
& & \Box \\
& & (\mathrm{minHit}(n) = \varnothing)\&\big(pass \to \mathbf{STOP}\big) \\
& & \Box \\
& & (k < j)\&\big(e : [n]^0 \to U_F(j, k+1, t(n,e))\big) \\
& & \Box \\
& & (k = j \wedge \mathrm{minHit}(n) \neq \varnothing)\&\big(\sqcap_{H \in \mathrm{minHit}(n)} (e : H \to pass \to \mathbf{STOP})\big)
\end{array}
$$

Check all traces up to length *j+1*

$$
\begin{aligned}
U_F(j) &= U_F(j, 0, \underline{n}) \\
U_F(j, k, n) &= \big(e : (\Sigma - [n]^0) \rightarrow fail \rightarrow \mathbf{STOP}\big) \\
&\quad \Box \\
&\quad (\mathrm{minHit}(n) = \varnothing) \& \big(pass \rightarrow \mathbf{STOP}\big) \\
&\quad \Box \\
&\quad (k < j) \& \big(e : [n]^0 \rightarrow U_F(j, k+1, t(n, e))\big) \\
&\quad \Box \\
&\quad (k = j \wedge \mathrm{minHit}(n) \neq \varnothing) \& \big(\sqcap_{H \in \mathrm{minHit}(n)} (e : H \rightarrow pass \rightarrow \mathbf{STOP})\big)
\end{aligned}
$$

When residing in node *n* of *P*'s transition graph, …

$$
\begin{aligned}
U_F(j) &= U_F(j,0,\underline{n}) \\
U_F(j,k,n) &= \big(e : (\Sigma - [n]^0) \to fail \to \mathbf{STOP}\big) \\
&\quad \Box \\
&\quad (\mathrm{minHit}(n) = \varnothing)\&\big(pass \to \mathbf{STOP}\big) \\
&\quad \Box \\
&\quad (k < j)\&\big(e : [n]^0 \to U_F(j,k+1,t(n,e))\big) \\
&\quad \Box \\
&\quad (k = j \wedge \mathrm{minHit}(n) \neq \varnothing)\&\big(\sqcap_{H \in \mathrm{minHit}(n)} (e : H \to pass \to \mathbf{STOP})\big)
\end{aligned}
$$

… offer any illegal event to *Q* which should not be accepted

Initials of node *n*

$$
\begin{aligned}
U_F(j) &= U_F(j, 0, \underline{n}) \\
U_F(j, k, n) &= \big(e : (\Sigma - [n]^0) \rightarrow fail \rightarrow \mathbf{STOP}\big) \\
&\quad \Box \\
&\quad (\mathrm{minHit}(n) = \varnothing) \& \big(pass \rightarrow \mathbf{STOP}\big) \\
&\quad \Box \\
&\quad (k < j) \& \big(e : [n]^0 \rightarrow U_F(j, k+1, t(n, e))\big) \\
&\quad \Box \\
&\quad (k = j \wedge \mathrm{minHit}(n) \neq \varnothing) \& \big(\sqcap_{H \in \mathrm{minHit}(n)} (e : H \rightarrow pass \rightarrow \mathbf{STOP})\big)
\end{aligned}
$$

... allow to stop with verdict PASS if *P* allows to refuse everything at node *n*

$$
\begin{aligned}
U_F(j) &= U_F(j, 0, \underline{n}) \\
U_F(j,k,n) &= \big(e : (\Sigma - [n]^0) \to fail \to \mathbf{STOP}\big) \\
&\quad \Box \\
&\quad (\mathrm{minHit}(n) = \varnothing)\&\big(pass \to \mathbf{STOP}\big) \\
&\quad \Box \\
&\quad (k < j)\&\big(e : [n]^0 \to U_F(j, k+1, t(n,e))\big) \\
&\quad \Box \\
&\quad (k = j \wedge \mathrm{minHit}(n) \neq \varnothing)\&\big(\sqcap_{H \in \mathrm{minHit}(n)} (e : H \to pass \to \mathbf{STOP})\big)
\end{aligned}
$$

... continue with any event which is admissible according to *n*'s initials, as long as the trace is still shorter than *j*. Continue with the successor node of *n* according to *P*'s transition function *t* ***(Back-to-Back Test Q against P)***

$$
\begin{aligned}
U_F(j) &= U_F(j, 0, \underline{n}) \\
U_F(j, k, n) &= \big(e : (\Sigma - [n]^0) \rightarrow fail \rightarrow \mathbf{STOP}\big) \\
&\quad \Box \\
&\quad (\mathrm{minHit}(n) = \varnothing) \& \big(pass \rightarrow \mathbf{STOP}\big) \\
&\quad \Box \\
&\quad (k < j) \& \big(e : [n]^0 \rightarrow U_F(j, k+1, t(n, e))\big) \\
&\quad \Box \\
&\quad (k = j \wedge \mathrm{minHit}(n) \neq \varnothing) \& \big(\sqcap_{H \in \mathrm{minHit}(n)} (e : H \rightarrow pass \rightarrow \mathbf{STOP})\big)
\end{aligned}
$$

… and check whether *Q* accepts an event from every minimal hitting set of node *n* without blocking, if the length of the trace is *j*

$$
\begin{aligned}
U_F(j) &= U_F(j, 0, \underline{n}) \\
U_F(j, k, n) &= \big(e : (\Sigma - [n]^0) \to fail \to \mathbf{STOP}\big) \\
&\quad \Box \\
&\quad (\mathrm{minHit}(n) = \varnothing) \& \big(pass \to \mathbf{STOP}\big) \\
&\quad \Box \\
&\quad (k < j) \& \big(e : [n]^0 \to U_F(j, k+1, t(n, e))\big) \\
&\quad \Box \\
&\quad (k = j \wedge \mathrm{minHit}(n) \neq \varnothing) \& \big(\sqcap_{H \in \mathrm{minHit}(n)} (e : H \to pass \to \mathbf{STOP})\big)
\end{aligned}
$$

PASS Criterion

- Q passes test $U_F(j)$, if and only if for every possible execution of
    - no FAIL event is ever produced,
    - the test always terminates with PASS

$$Q \ \underline{pass} \ U_F(j) \equiv (pass \rightarrow \textbf{STOP}) \sqsubseteq_F (Q \parallel_\Sigma U_F(j)) \backslash \Sigma$$

Complete Testing Assumption

- There exists a constant $c \geq 1$, such that every possible behaviour of the SUT, when running in parallel to test case *U,* is exhibited within *c* executions of *U*

Completeness Result

Theorem. Let P be a non-terminating, divergence-free CSP process over alphabet Σ whose normalised transition graph $G(P)$ has p states. Define fault domain $\mathcal{D}$ as the set of all divergence-free CSP processes over alphabet Σ, whose transition graph has at most q states with $q \geq p$. Then the test suite

$$TS_F = \{U_F(j) \mid 0 \leq j < pq\}$$

is complete with respect to $\mathcal{F} = (P, \sqsubseteq_F, \mathcal{D})$.

Complexity Considerations

Maximal Number of Test Executions Required

Theorem. The maximal number of test executions to be performed using the complete test suite $TS_F = \{U_F(j) \mid 0 \leq j < pq\}$ created from P is of order

$$O(\binom{n}{\lfloor \frac{n}{2} \rfloor} \cdot n^{pq-1}) \quad \text{with } n = |\Sigma|.$$

For processes P satisfying $(s, \Sigma) \notin \text{failures}(P)$ for all traces s, the reachable precise upper bound is given by

$$\binom{n}{\lfloor \frac{n}{2} \rfloor} \cdot \frac{1 - n^{pq}}{1 - n} \quad \text{with } n = |\Sigma|.$$

Maximal Number of Test Executions Required

Theorem. The maximal number of test executions to be performed using the complete test suite $TS_F = \{U_F(j) \mid 0 \leq j < pq\}$ created from P is of order

$$O(\binom{n}{\lfloor \frac{n}{2} \rfloor} \cdot n^{pq-1}) \quad \text{with } n = |\Sigma|.$$

For processes P satisfying $(s, \Sigma) \notin \text{failures}(P)$ for all traces s, the reachable precise upper bound is given by

$$\binom{n}{\lfloor \frac{n}{2} \rfloor} \cdot \frac{1 - n^{pq}}{1 - n} \quad \text{with } n = |\Sigma|.$$

Maximal number of hitting sets to be tested at the end of each test execution

Why we Cannot use Shorter Traces

Theorem. Let $2 \leq p, q \in \mathbb{N}$. Then there exists a reference process P and an implementation process Q with the following properties.

1. $G(P)$ has p states.
2. $G(Q)$ has q states.
3. $P \not\sqsubseteq_T Q$, and therefore, also $P \not\sqsubseteq_F Q$.
4. $\forall s \in \text{traces}(Q) : \#s < pq \implies s \in \text{traces}(P)$.
5. $Q\ conf\ P$.

Discussion

Discussion

- **Could the test effort be further reduced?**
  - We know that maximal length of traces and number of probes *H* cannot be reduced without losing completeness
  - Translate results about adaptive state counting methods from FSMs to CSP

R. M. Hierons, Testing from a nondeterministic finite state machine using adaptive state counting, IEEE Trans. Computers 53 (10) (2004) 1330–1342. doi:10.1109/TC.2004.85

Discussion

- **Is it such a good idea to check for *refinement*?**
    - From complete methods for FSMs we know that **complete equivalence checking** can be performed with much shorter traces**:** **maximal length $p+q-1$** (instead of $pq - 1$ as required for refinement checking)
    - Alternative approach to be preferred:
        - refine original model and check correctness by model checker FDR4
        - Stop refinement as soon as implementation can be required to be equivalent to last refinement
        - Then **test for failures equivalence**

Discussion

- **Implications for CSP model checking**
  - As an alternative to checking $P \sqsubseteq_F Q$, could it be effective to use an estimate for *q* and perform **concurrent** checks

$$(pass \rightarrow \textbf{STOP}) \sqsubseteq_F (Q \parallel_\Sigma U_F(j)) \backslash \Sigma, \quad j = 0, \ldots, pq - 1$$

Acknowledgements

The authors would like to thank Bill Roscoe and Thomas Gibson-Robinson for their advice on using the FDR4 model checker and for very helpful discussions concerning the potential implications of this paper in the field of model checking. We are also grateful to Li-Da Tong from National Sun Yat-sen University, Taiwan, for suggesting the applicability of Sperner's Theorem in the context of the work presented here. Moreover, we thank Adenilso Simao for several helpful suggestions. The work of Ana Cavalcanti is funded by the Royal Academy of Engineering and UK EPSRC Grant EP/R025134/1.


PUSH
STOP ARRET

Appendix.
Three
Mathematical Tools

Product Graphs
Minimal Hitting sets
Sperner Families

Product Graphs

$$\begin{aligned}
G_1 \times G_2 &= (N_1 \times N_2, (\underline{n}_1, \underline{n}_2), t : (N_1 \times N_2) \times \Sigma \not\rightarrow (N_1 \times N_2)) \\
\text{dom } t &= \{((n_1, n_2), e) \in (N_1 \times N_2) \times \Sigma | (n_1, e) \in \text{dom } t_1 \wedge (n_2, e) \in \text{dom } t_2\} \\
t((n_1, n_2), e) &= (t_1(n_1, e), t_2(n_2, e)) \text{ for } ((n_1, n_2), e) \in \text{dom } t
\end{aligned}$$

Product Graphs

Graph nodes are product of nodes of each operand

$$
\begin{aligned}
G_1 \times G_2 &= (N_1 \times N_2, (\underline{n}_1, \underline{n}_2), t : (N_1 \times N_2) \times \Sigma \not\rightarrow (N_1 \times N_2)) \\
\text{dom } t &= \{((n_1, n_2), e) \in (N_1 \times N_2) \times \Sigma | (n_1, e) \in \text{dom } t_1 \wedge (n_2, e) \in \text{dom } t_2\} \\
t((n_1, n_2), e) &= (t_1(n_1, e), t_2(n_2, e)) \text{ for } ((n_1, n_2), e) \in \text{dom } t
\end{aligned}
$$

Product Graphs

Initial state is pair of operand's initial states

$$
\begin{aligned}
G_1 \times G_2 &= (N_1 \times N_2, (\underline{n}_1, \underline{n}_2), t : (N_1 \times N_2) \times \Sigma \not\rightarrow (N_1 \times N_2)) \\
\text{dom } t &= \{((n_1, n_2), e) \in (N_1 \times N_2) \times \Sigma | (n_1, e) \in \text{dom } t_1 \wedge (n_2, e) \in \text{dom } t_2\} \\
t((n_1, n_2), e) &= (t_1(n_1, e), t_2(n_2, e)) \text{ for } ((n_1, n_2), e) \in \text{dom } t
\end{aligned}
$$

Product Graphs

Transition function allows transition labelled by event *e* iff both "local" transition functions ***t_1*, *t_2*** allow for this transition from their respective source states

$$
\begin{aligned}
G_1 \times G_2 &= (N_1 \times N_2, (\underline{n}_1, \underline{n}_2), t : (N_1 \times N_2) \times \Sigma \not\rightarrow (N_1 \times N_2)) \\
\text{dom } t &= \{((n_1, n_2), e) \in (N_1 \times N_2) \times \Sigma | (n_1, e) \in \text{dom } t_1 \wedge (n_2, e) \in \text{dom } t_2\} \\
t((n_1, n_2), e) &= (t_1(n_1, e), t_2(n_2, e)) \text{ for } ((n_1, n_2), e) \in \text{dom } t
\end{aligned}
$$

Lemma on Product Graph

Lemma. If G_1 has p states and G_2 has q states, then $G_1 \times G_2$ has at most pq states, and every reachable state of $G_1 \times G_2$ can be reached by a trace of maximal length $pq - 1$.

Lemma on Product Graph

Lemma. If G_1 has p states and G_2 has q states, then $G_1 \times G_2$ has at most pq states, and every reachable state of $G_1 \times G_2$ can be reached by a trace of maximal length $pq - 1$.

Apply this lemma as follows

- G_1 is the normalised transition graph of reference process *P*
- G_2 is the (unknown) normalised transition graph of SUT with behaviour *Q*
- **Hypothesis.** G_2 has at most $q \geq p$ states
- Suppose that *Q* exhibits faulty behaviour at some graph node n_2. Then
  - either this state can be reached by a trace of *P* with length < pq,
  - or *Q* refuses to continue a shorter trace with an event which should not be refused according to reference process *P*

Minimal Hitting Sets

- Given a finite universe Σ, and a collection of subsets $\{A_1,\ldots,A_k\}$, a subset ***H* ⊆ Σ is called a hitting set** of $\{A_1,\ldots,A_k\}$, if and only if $H \cap A_i \neq \{\,\}$ for all $i = 1,\ldots,k$

- A hitting set *H* is called **minimal**, if no true subset of *H* is a hitting set of $\{A_1,\ldots,A_k\}$

Minimal Hitting sets of Minimal Acceptances Characterise *conf*

Lemma. Let P, Q be two finite-state CSP processes. For each $s \in \text{traces}(P)$, let $\text{minHit}(P/s)$ denote the collection of all minimal hitting sets of $\text{minAcc}(P/s)$. Then the following statements are equivalent.

1. $Q \ conf \ P$
2. For all $s \in \text{traces}(P) \cap \text{traces}(Q)$ and $H \in \text{minHit}(P/s)$, H is a (not necessarily minimal) hitting set of $\text{minAcc}(Q/s)$.

Sperner Families

- **Sperner Family**. A collection of subsets of finite universe Σ, such that no pair of distinct sets in the family are in subset relation

$$\{A_1, \ldots, A_k\} \subseteq 2^{\Sigma}, \quad \forall i \neq j : A_i \nsubseteq A_j \wedge A_j \nsubseteq A_i$$

- **Sperner's Theorem**. The cardinality of a Sperner family is bounded by

$$\binom{n}{\lfloor \frac{n}{2} \rfloor} \quad with \quad n = |\Sigma|$$

Sperner Families

- **Sperner Family**. A collection of subsets of finite universe Σ, such that no pair of distinct sets in the family are in subset relation

This maximum is reached

- For even *n*: all subsets of Σ with cardinality *n/2*
- For odd *n* : all subsets of Σ with cardinality *(n-1)/2*
- For odd *n* : all subsets of Σ with cardinality *(n+1)/2*

- **Sperner's Theorem**. The cardinality of a Sperner family is bounded by

$$\binom{n}{\lfloor \frac{n}{2} \rfloor} \quad with \quad n = |\Sigma|$$

E. Sperner, Ein Satz über Untermengen einer endlichen Menge, Mathematische Zeitschrift 27 (1) (1928) 544–548. doi:10.1007/ BF01171114.

Sperner Families in our Context

- Maximal refusals of a process state
- Minimal acceptances of a process state
- Minimal hitting sets

Appendix.
Semantics of CSP, Refusals and Acceptances

Overview

- A new result: **finite** complete test suites for CSP conformance relations
  - **traces refinement**
  - **failures refinement**
- Complexity bounds
- Presentation of methods that are universally applicable for arbitrary formalisms

CSP

Nondeterministic communicating sequential processes over finite alphabets

Deadlock process **STOP**

Prefixing with events $a \to b \to c \to$ **STOP**

Process equations with recursion

$$P = a \to Q$$
$$Q = b \to P$$

External choice $P = (a \to P \,\Box\, b \to c \to P)$

Internal choice $P = (a \to P \sqcap b \to c \to P)$

Concurrent processes synchronised over set of events $P \; [\,|\,\{a, b, c\}\,|\,] \; Q$

CSP Traces Semantics

- traces(P) – language generated by CSP process P
- Prefix-closed

$$\textbf{traces}(a \rightarrow b \rightarrow \textbf{STOP}) = \{\varepsilon, a, a\,.\,b\}$$

- Denotational traces semantics provides compositional rules about how to compute the traces of a composed processes, provided that the traces of the operands are known

$$\textbf{traces}(P \sqcap Q) = \textbf{traces}(P) \cup \textbf{traces}(Q)$$

CSP Failures Semantics

- **Refusal.** A set of events that may be refused in a certain process state *P/s* (= *P*, after having run through trace *s*)
  - Refusals are subset-closed
  - All refusals of a process state can be calculated from its maximal refusals, due to subset closure
- **Failure.** A pair *(s,R)*, such that *R* is a refusal of *P/s*
- **failures(P)** can be calculated via compositional rules of the denotational semantics

Example

Rule

$$\textbf{failures}(\textbf{STOP}) = \{(\varepsilon, R) \mid R \subseteq \Sigma\}$$

Rule

$$\textbf{failures}(a \rightarrow P) = \{(\varepsilon, R) \mid R \subseteq \Sigma - \{a\}\} \cup \{(a \,.\, s, R) \mid (s, R) \in \textbf{failures}(P)\}$$

Conclusion

$$\textbf{failures}(a \rightarrow \textbf{STOP}) = \{(\varepsilon, R) \mid R \subseteq \Sigma - \{a\}\} \cup \{(a, R) \mid R \subseteq \Sigma\}$$

Conformance Relations in CSP

- **Trace refinement** $P \sqsubseteq_T Q \equiv \textbf{traces}(Q) \subseteq \textbf{traces}(P)$
- **Trace equivalence** $P =_T Q \equiv \textbf{traces}(Q) = \textbf{traces}(P)$
- **Failures refinement** $P \sqsubseteq_F Q \equiv \textbf{failures}(Q) \subseteq \textbf{failures}(P)$
- **Failures equivalence** $P =_F Q \equiv \textbf{failures}(Q) = \textbf{failures}(P)$

Auxiliary Conformance Relation

$$Q\ conf\ P \equiv \forall s \in \textbf{traces}(P) \cap \textbf{traces}(Q) : \textbf{Ref}(Q/s) \subseteq \textbf{Ref}(P/s)$$

Lemma. $P \sqsubseteq_F Q \Leftrightarrow P \sqsubseteq_T Q \wedge Q\ conf\ P$

A. Cavalcanti, M. Gaudel, Testing for refinement in CSP, in: M. J. Butler, M. G. Hinchey, M. M. Larrondo-Petrie (Eds.), Formal Methods and Software Engineering, 9th International Conference on Formal Engineering Methods, ICFEM 2007, Boca Raton, FL, USA, November 14-15, 2007, Proceedings, Vol. 4789 of Lecture Notes in Computer Science, Springer, 2007, pp. 151–170. doi: 10.1007/978-3-540-76650-6\ _10.

J. Tretmans. A formal approach to conformance testing. PhD thesis, University of Twente, Enschede, The Netherlands, 1992.

Acceptances vs. Refusals

- A set of events *A* is an **acceptance** of process state, if *A* is the complement of a refusal R in this state

$A = \Sigma - R$

- A **minimal acceptance** is a complement of a maximal refusal

- **Saturation property**: *A* is an acceptances of a process state *P/s,* if *A* is
  - a superset of some minimal acceptance, and
  - a subset of the state's initials $[P/s]^0$

$A_{min} \subseteq A \subseteq [P/s]^0$