Abstracting from Failure Probabilities

Jan Jürjens

Computing Laboratory, University of Oxford

jan@comlab.ox.ac.uk

http://www.jurjens.de/jan

Formal methods for dependability

Safety-critical systems:
Use formal methods to demonstrate dependability.

Abstract from failure probabilities to keep treatment feasible.

Assume failure masked perfectly by fault-tolerance mechanisms, considered in isolation of system.

Assume failure distributions uncorrelated,
constant over time or use etc.

Abstracting from Failure ?

Under which assumptions and to what extent is this sound ?

- obtain concrete information on dependability of overall system, given dependability of fault-tolerance components

- do without usual assumptions (e.g. history-independence of failure probabilities)

Concrete estimates useful: no more redundancy than necessary (increased replication $\leadsto$ increased cost, decreased performance, failures due to increase in complexity).

This work

Within Focus (M. Broy) define specification language interpreted at two levels of abstraction:

- abstract model: reason about system by treating fault-tolerance components as "black boxes",
- probabilistic model: failure probabilities retained.

Determine degree of faithfulness of abstract model wrt. probabilistic one.

Precise probabilities not always available.
Reason qualitatively: Define dependability asympotically as function of safety parameter.

Process model

Nondeterministic, concurrently executing processes.

Communication asynchronous:
receiver cannot prevent transmission of value.

Process P: collection of synchronously executing programs;
communicate through channels.

For each output channel c, a program p_c computes
output on c from data on P's input channels.

Local state through feedback channels, use for iteration.

Specification language: Expressions

Data sent over channels: formal expressions defined by

$E ::=$	expression
$\quad \perp$	error value
$\quad x$	variable ($x \in \mathbf{Var}$)
$\quad c$	input value ($c \in \mathbf{Channels}$)
$\quad d$	data value ($d \in \mathcal{D}$)
$\quad E_1 :: E_2$	concatenation

Channel name c refers to input on c at given point in time.

Empty expression ε: absence of value. Error value $\perp$: failure.

Specification language: Programs

(Nondeterministic) programs defined by:

$p ::=$	programs
$\quad E$	output of expression
$\quad either\ p\ or\ p'$	nondeterminism
$\quad if\ E = E'\ then\ p\ else\ p'$	conditional
$\quad case\ E\ of\ x :: y\ do\ p\ else\ p'$	break up list

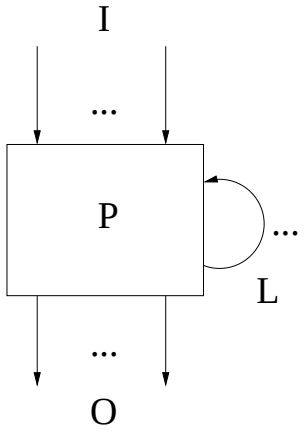
x and y: bound variables.

Program closed: contains no unbound variables.

Specification language: Processes

Process: of form $P = (I, O, L, (p_c)_{c \in \tilde{O}})$ where

- $I \subseteq \mathbf{Channels}$ input channels,
- $O \subseteq \mathbf{Channels}$ output channels,
- $L \subseteq \mathbf{Channels}$ local channels,
- each p_c closed program with input channels in $\tilde{I} \stackrel{\text{def}}{=} I \cup L$.



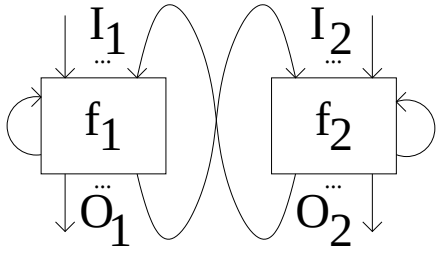
Stream-processing functions

$\mathbf{Stream}_C \stackrel{\text{def}}{=} (\mathbf{CExp}^\omega)^C$: set of C-indexed tuples of sequences of closed expressions.

$\vec{s}(c)$: sequence of expressions at channel c.
Process P defines $[\![P]\!] : \mathbf{Stream}_I \to (\mathbf{Stream}_O)^{\mathcal{C}(P)}$.

Composition of stream-processing functions
$f_i : \mathbf{Stream}_{I_i} \to (\mathbf{Stream}_{O_i})^{D_i}$:

$$f_1 \otimes f_2 : \mathbf{Stream}_I \to (\mathbf{Stream}_O)^{D_1 \times D_2}$$



$(I = (I_1 \cup I_2) \setminus (O_1 \cup O_2), O = (O_1 \cup O_2) \setminus (I_1 \cup I_2))$.

For $\vec{s} \in \mathbf{Stream}_I$, (d_1, d_2)-component of $f_1 \otimes f_2(\vec{s})$ is $\vec{t}\downarrow_O$, where $\vec{t}$ satisfies $\vec{t}\downarrow_I = \vec{s}\downarrow_I$ and $\vec{t}\downarrow_{O_i} = f_i(\vec{s}\downarrow_{I_i})_{d_i}$.

Abstract Semantics (1)

For closed deterministic program p, define $[p](\vec{M}) \in \mathbf{CExp}$:

$$[E](\vec{M}) = E(\vec{M}) \qquad E \in \mathbf{CExp}$$

$$[\mathit{if}\ E = E'\ \mathit{then}\ p\ \mathit{else}\ p'](\vec{M}) = [p](\vec{M}) \qquad \text{if } [E](\vec{M}) = [E'](\vec{M})$$

$$[\mathit{if}\ E = E'\ \mathit{then}\ p\ \mathit{else}\ p'](\vec{M}) = [p'](\vec{M}) \qquad \text{if } [E](\vec{M}) \neq [E'](\vec{M})$$

$$[\mathit{case}\ E\ \mathit{of}\ x :: y\ \mathit{do}\ p\ \mathit{else}\ p'](\vec{M}) = [p[h/x, t/y]](\vec{M})$$
$$\text{if } [E](\vec{M}) = h :: t \text{ with } h \in \{\bot\} \cup \mathbf{Var} \cup \mathcal{D} \cup \tilde{I}$$

$$[\mathit{case}\ E\ \mathit{of}\ x :: y\ \mathit{do}\ p\ \mathit{else}\ p'](\vec{M}) = [p'](\vec{M}) \qquad \text{if } [E](\vec{M}) = \varepsilon$$

$[p](\vec{M})$: result from running p once on input $\vec{M}$.

Abstract Semantics (2)

Define $[p](\vec{s}) \in (\mathbf{Stream}_c)^{\mathcal{C}(p)}$ for nondeterministic programs p:

For each deterministic component p' of p,
let p'-component of $[p](\vec{s})$ be $\vec{t} \in \mathbf{Stream}_c$ with

- $\vec{t}_0 = [p'](\varepsilon, \ldots, \varepsilon)$ and
- $\vec{t}_{n+1} = [p'](\vec{s}_n)$.

Process $P = (I, O, L, (p_c)_{c \in \tilde{O}})$ interpreted as
$[\![P]\!] \stackrel{\text{def}}{=} \otimes_{c \in \tilde{O}} [p_c] : \mathbf{Stream}_I \to (\mathbf{Stream}_O)^{\mathcal{C}(P)}$.

Fault-Tolerance

Assume same failure distribution and
replication mechanism Π throughout system.

Model Π as family of distributions τ_η
(with safety parameter $\eta \in \mathbb{N}$) on $\mathbf{Failures} \stackrel{\text{def}}{=} \{0,1\}^{\mathbb{N}}$
(1: unmasked failure).

Probability that n^{th} access raises failure: $\Pr[\vec{r} \leftarrow \tau_\eta : \vec{r}_n = 1]$.

Probabilistic Semantics (1)

For closed deterministic program p define $[p]_{\vec{r}}(\vec{M}) \in \mathbf{CExp}$:

$[E]_{0.\vec{r}}(\vec{M}) = E(\vec{M})$ for $E \in \mathbf{CExp}$

$[E]_{1.\vec{r}}(\vec{M}) = \bot$

$[\mathit{if}\ E = E'\ \mathit{then}\ p\ \mathit{else}\ p']_{r.\vec{r}}(\vec{M}) = [p]_{\vec{r}}(\vec{M})$ if $[E]_{r.\vec{r}}(\vec{M}) = [E']_{r.\vec{r}}(\vec{M})$

$[\mathit{if}\ E = E'\ \mathit{then}\ p\ \mathit{else}\ p']_{r.\vec{r}}(\vec{M}) = [p']_{\vec{r}}(\vec{M})$ if $[E]_{r.\vec{r}}(\vec{M}) \neq [E']_{r.\vec{r}}(\vec{M})$

$[\mathit{case}\ E\ \mathit{of}\ x :: y\ \mathit{do}\ p\ \mathit{else}\ p']_{r.\vec{r}}(\vec{M}) = [p']_{\vec{r}}(\vec{M})$ if $[E]_{r.\vec{r}}(\vec{M}) = \varepsilon$

$[\mathit{case}\ E\ \mathit{of}\ x :: y\ \mathit{do}\ p\ \mathit{else}\ p']_{r.\vec{r}}(\vec{M}) = [p[h/x, t/y]]_{\vec{r}}(\vec{M})$

if $[E]_{r.\vec{r}}(\vec{M}) = h :: t$ where $h \in \{\bot\} \cup \mathbf{Var} \cup \mathcal{D} \cup \tilde{I}$.

$[p]_{\vec{r}}(\vec{M})$: result from running p once with initial values $\vec{M}$ and failure occurences $\vec{r}$.

Probabilistic Semantics (2)

Non-deterministic p: For deterministic component p' of p, let p'-component of $[p]_{\vec{r}}(\vec{s})$ be $\vec{t} \in \mathbf{Stream}_c$ with

- $\vec{t}_0 = [p']_{r^0}(\varepsilon, \ldots, \varepsilon)$ and
- $\vec{t}_{n+1} = [p']_{r^{n+1}}(\vec{s}_n)$.

$P = (I, O, L, (p_c)_{c \in \tilde{O}})$ interpreted as $[\![P]\!]_{\vec{r}} \stackrel{\text{def}}{=} \bigotimes_{c \in \tilde{O}} [p_c]_{r^c}$.

$[\![P]\!]_\Pi \stackrel{\text{def}}{=} \{[\![P]\!]_{\vec{r}} : \vec{r} \leftarrow \tau_\eta\}$ for replication mechanism $\Pi = (\tau_\eta)_\eta$.

Faithfulness of Abstract Model (1)

Theorem

Suppose $\Pi = (\tau_\eta)_\eta$ is $p(\eta, t)$-safe replication mechanism:
for any $(\eta, t) \in \mathbb{N} \times \mathbb{N}$ have $\mathsf{Pr}\,[\vec{r} \leftarrow \tau_\eta : \vec{r}_t = 1] \leq p(\eta, t)$.

Then P is $\delta(\eta, l)$-safe for $\delta(\eta, l) \stackrel{\text{def}}{=} \sum_{i=1,\ldots,l \cdot n(P)} p(\eta, i)$:
for any deterministic component P' of P, any $(\eta, l) \in \mathbb{N} \times \mathbb{N}$,
and any $\vec{s} \in \mathbf{Stream}_I$ of length up to l
have $\mathsf{Pr}\,[\vec{r} \leftarrow \tau_\eta : [\![P']\!]_{\vec{r}}(\vec{s}) \neq [\![P']\!](\vec{s})] \leq \delta(\eta, l)$.

Optimal in absence of further assumptions on Π.

Faithfulness of Abstract Model (2)

Definition Replication mechanism $\Pi = (\tau_\eta)_\eta$ history-independent: for all bit-sequences $\vec{r}_1, \vec{r}_2 \in \{0,1\}^l$ and each $b \in \{0,1\}$ have

$$\frac{\Pr\left[\vec{r} \leftarrow \tau_\eta : \vec{r}\lfloor_{l+1} = \vec{r}_1.b\right]}{\Pr\left[\vec{r} \leftarrow \tau_\eta : \vec{r}\lfloor_{l} = \vec{r}_1\right]} = \frac{\Pr\left[\vec{r} \leftarrow \tau_\eta : \vec{r}\lfloor_{l+1} = \vec{r}_2.b\right]}{\Pr\left[\vec{r} \leftarrow \tau_\eta : \vec{r}\lfloor_{l} = \vec{r}_2\right]}$$

where $\vec{r}\lfloor_n$ is the prefix of length n of $\vec{r}$.

May still depend on time.

Theorem

$p(\eta, t)$-safe history-independent replication mechanism Π.

Then P is $\delta(\eta, l)$-safe for $\delta(\eta, l) \stackrel{\text{def}}{=} 1 - \Pi_{i=1,\ldots,l \cdot n(P)}(1 - p(\eta, i))$.

Example: Crash/performance failure semantics

Crash/performance failure semantics: component may crash or deliver data after time limit, but partially correct.

Replication mechanism $\Pi_{c/p}$: group of η replicated components; fastest determines output. Suppose component failure probability p (independent from each other and from history).

Fact $\Pi_{c/p}$ is $p(\eta, t)$-safe replication mechanism for $p(\eta, t) \stackrel{\text{def}}{=} p^{\eta}$.

Suppose probability that component fails at t^{th} access is $p \cdot t/(t+1)$, gives replication mechanism $\Pi'_{c/p}$.

Fact $\Pi'_{c/p}$ is $p(\eta, t)$-safe $p(\eta, t) \stackrel{\text{def}}{=} (p \cdot t/(t+1))^{\eta}$.

Example: Unbounded buffer (1)

$P = (\{c\}, \{d\}, \{l\}, (p_d, p_l))$ unbounded FIFO buffer:

stores received data; outputs on request:

$$p_l \stackrel{\text{def}}{=} \mathit{if}\ c = \text{request}\ \mathit{then}\ (\mathit{case}\ l\ \mathit{of}\ h :: t\ \mathit{do}\ t\ \mathit{else}\ \varepsilon)$$
$$\mathit{else}\ l :: c$$
$$p_d \stackrel{\text{def}}{=} \mathit{if}\ c = \text{request}\ \mathit{then}\ (\mathit{case}\ l\ \mathit{of}\ h :: t\ \mathit{do}\ h\ \mathit{else}\ \varepsilon)$$
$$\mathit{else}\ \varepsilon$$

Example: Unbounded buffer (2)

Want dependable implementation using hardware with crash/performance failure semantics $\Pi_{c/p}$.

Proposition

P is $\delta(\eta, l)$-safe for $\delta(\eta, l) \stackrel{\text{def}}{=} 1 - (1 - p^{\eta})^{l \cdot n(P)} = 1 - (1 - p^{\eta})^{4 \cdot l}$.

Now consider $\Pi'_{c/p}$ (time-dependent failures):

Proposition

P is $\delta(\eta, l)$-safe for $\delta(\eta, l) \stackrel{\text{def}}{=} 1 - \Pi_{i=1,\ldots,l \cdot n(P)}(1 - (p \cdot i/(i+1))^{\eta})$.

Here time-independence violated.

Conclusion

Showed formally

- under which assumptions and
- to what extent

abstracting from failure probabilities is sound.

Obtained probability bounds on failure of overall system,
given dependability of fault-tolerance components.

Can do without some of usual assumptions.

Can reason qualitatively (asymptotically using safety-parameters).

Future Work

Apply to real-life examples.

Assumed uniform failure semantics and replication mechanism.
Allow more flexibility (also failure correlation).

More expensive to provide highly dependable components,
but cheaper to handle failure behaviours when accessing them
(group management: up to 80% of total throughput).
$\rightsquigarrow$ extend our approach with performance aspects.

Extend approach to other formalisms.

Deterministic components

Deterministic components of program p:
substitute subconstruct $either\ p_1\ or\ p_2$ by p_1 or p_2.

$p \stackrel{\text{def}}{=} if\ E = \perp\ then\ (either\ 0\ or\ 1)\ else\ (either\ 0\ or\ 1)$ has deterministic components $if\ E = \perp\ then\ i\ else\ j$ with $i, j \in \{0, 1\}$.

Data access bounds

For program p, define bound $n(p)$ on number of data accesses during one iteration of execution:

- $n(E) = 1$ for an expression $E \in \mathbf{Exp}$
- $n(\mathit{either}\ p\ \mathit{or}\ p') = \max(n(p), n(p'))$
- $n(\mathit{if}\ E = E'\ \mathit{then}\ p\ \mathit{else}\ p') = \max(n(p), n(p')) + 2$
- $n(\mathit{case}\ E\ \mathit{of}\ x :: y\ \mathit{do}\ p\ \mathit{else}\ p') = \max(n(p), n(p'))\ +1$

For process $P = (I, O, L, (p_c)_{c \in \tilde{O}})$ define $n(P) = \sum_{c \in \tilde{O}} n(p_c)$.