Reasoning about Reconfigurations of Distributed Systems

Emma Ahrens, Joost-Pieter Katoen (RWTH Aachen), Marius Bozga, Radu Iosif (VERIMAG Grenoble),
Why is reconfiguration needed?
Why is reconfiguration needed?
Why is reconfiguration needed?
Why is reconfiguration needed?
Why is reconfiguration needed?
Why is reconfiguration needed?
Why is reconfiguration needed?
Types of Dynamic Reconfiguration

- **Internal vs external** initiation of architectural changes
  - self-managing systems have internal initiation based on guards

- **Basic** (component/interaction addition/removal) and **composite** (sequencing, choice, iteration) operations

- **Centralized vs distributed** management
  - centralized (sequential) management is simpler to implement and supported by the majority of dynamic reconfiguration languages
  - distributed (parallel) management is more efficient and realistic but more challenging to model and reason about
Components and Interactions

- An architecture designates the set of components (network nodes) and interactions (connectors).

- A component encapsulates behavior (set of event traces) by means of a well-defined interface (= set of communication ports).

- We currently abstract behavior by finite-state machines.
Components and Interactions

- An interaction connects one or more components
  - interactions involving one component correspond to local actions
  - communication is considered lossless and instantaneous
Components and Interactions

- An interaction connects one or more components
  - interactions involving one component correspond to local actions
  - communication is considered lossless and instantaneous
Components and Interactions

- An interaction connects one or more components
  - interactions involving one component correspond to local actions
  - communication is considered lossless and instantaneous
Reconfiguration: what can possibly go wrong?
Reconfiguration: what can possibly go wrong?
Reconfiguration: what can possibly go wrong?

disconnect(T,y,z);
Reconfiguration: what can possibly go wrong?

disconnect(T,y,z);
disconnect(T,x,y);
Reconfiguration: what can possibly go wrong?

disconnect(T,y,z);
disconnect(T,x,y);
delete(S,y);
Reconfiguration: what can possibly go wrong?

disconnect(T,y,z);
disconnect(T,x,y);
delete(T,y);
connect(T,x,z);
Reconfiguration: what can possibly go wrong?

disconnect(T,y,z);
disconnect(T,x,y);
delete(T,y);
connect(T,x,z);
deadlock
Proving Correctness of a System after Reconfiguration

1. Resource logic-based assertion language for describing infinite sets of configurations
2. Hoare-style proof calculus for reasoning about reconfiguration programs
3. Sound push-button technique for proving safety of a set of configurations
What are the Configurations?

A configuration is an architecture with a snapshot of the states of each component.
Havoc vs Reconfiguration Actions
Havoc vs Reconfiguration Actions
Havoc vs Reconfiguration Actions
Havoc vs Reconfiguration Actions
Havoc vs Reconfiguration Actions

disconnect(T,x,y)
Havoc vs Reconfiguration Actions

disconnect(T,x,y)
**Havoc vs Reconfiguration Actions**

`disconnect(T,x,y)`
Havoc vs Reconfiguration Actions

disconnect(T,x,y)

disconnect(T,y,z)
Havoc vs Reconfiguration Actions

disconnect(T,x,y)

disconnect(T,y,z)
Havoc vs Reconfiguration Actions

disconnect(T,x,y)

disconnect(T,y,z)
Reconfiguration Programs are Infinite-state Systems

- Transition system with unbounded number of configurations:
  - new components can be added, yielding increasingly complex reachability graphs
Reconfiguration Programs are Infinite-state Systems

- Transition system with unbounded number of configurations:
  - new components can be added, yielding increasingly complex reachability graphs

- Two orthogonal types of actions that interleave:
  - **reconfiguration actions** change the architecture of a system
  - **havoc actions** are state changes caused by firing interactions
A Logic of Configurations (CL)

emp the empty architecture
C^q(x) a single component of type C in state q and no interactions
I(x_1, ..., x_n) a single interaction of type I (of arity n) and no components
ϕ_1 * ϕ_2 union of pointwise disjoint architectures
A Logic of Configurations (CL)

\[ \text{emp} \]
the empty architecture

\[ C^q(x) \]
a single component of type C in state q and no interactions

\[ I(x_1, \ldots, x_n) \]
a single interaction of type I (of arity n) and no components

\[ \phi_1 \ast \phi_2 \]
union of pointwise disjoint architectures

\[ S^{\text{token}}(x) \ast T(x, y) \]
A Logic of Configurations (CL)

emp
C_q(x)
I(x_1, ...., x_n)
\( \phi_1 \ast \phi_2 \)

the empty architecture
a single component of type C in state q and no interactions
a single interaction of type I (of arity n) and no components
union of pointwise disjoint architectures

\[ S^{\text{token}}(x) \ast T(x,y) \]
\[ S^{\text{hole}}(y) \ast T(y,z) \]
A Logic of Configurations (CL)

\( \text{emp} \) the empty architecture

\( C^q(x) \) a single component of type \( C \) in state \( q \) and no interactions

\( I(x_1, \ldots, x_n) \) a single interaction of type \( I \) (of arity \( n \)) and no components

\( \Phi_1 \ast \Phi_2 \) union of pointwise disjoint architectures

\( S^{\text{token}}(x) \ast T(x, y) \)

\( S^{\text{hole}}(y) \ast T(y, z) \)

\( S^{\text{hole}}(z) \ast T(z, x) \)
A Logic of Configurations (CL)

emp
the empty architecture

$C^q(x)$
a single component of type C in state q and no interactions

$I(x_1, \ldots, x_n)$
a single interaction of type I (of arity n) and no components

$\phi_1 \ast \phi_2$
union of disjoint architectures

$S^\text{token}(x) \ast T(x,y) \ast S^\text{hole}(y) \ast T(y,z) \ast S^\text{hole}(z) \ast T(z,x)$
A Logic of Configurations (CL)

emp  the empty architecture
C^q(x) a single component of type C in state q and no interactions
l(x_1, ..., x_n) a single interaction of type l (of arity n) and no components
ϕ_1 * ϕ_2 separating conjunction (union of disjoint architectures)
ϕ_1 ∧ ϕ_2 boolean conjunction
¬ϕ negation
∃x . ϕ existential quantification
CL is a Resource Logic

emp the empty architecture
C_q(x) a single component of type C in state q and no interactions
l(x_1, ..., x_n) a single interaction of type l (of arity n) and no components
\phi_1 \ast \phi_2 separating conjunction (union of disjoint architectures)
\phi_1 \land \phi_2 boolean conjunction

C_{q_1}(x) \ast C_{q_2}(y) implies x \neq y
C_{q_1}(x) \land C_{q_2}(y) implies x = y (also q_1 = q_2)
CL is a Resource Logic

- **emp**: the empty architecture
- **$C^q(x)$**: a single component of type C in state q and no interactions
- **$I(x_1, \ldots, x_n)$**: a single interaction of type I (of arity n) and no components
- **$\phi_1 \ast \phi_2$**: separating conjunction (union of disjoint architectures)
- **$\phi_1 \land \phi_2$**: boolean conjunction

**Formulas**

- $C^q_1(x) \ast C^q_2(y)$ implies $x \neq y$
- $C^q_1(x) \land C^q_2(y)$ implies $x = y$ (also $q_1 = q_2$)
- $I(x,y) \ast I(z,v)$ implies $x \neq z$ or $y \neq v$
- $I(x,y) \land I(z,v)$ implies $x = z$ and $y = v$
Adding Inductive Definitions

\[ \text{Ring}_{h,t}() \]
Adding Inductive Definitions

Ring_{h,t}() \leftarrow \exists y_1 \exists y_2 . \text{Chain}_{h,t}(y_1, y_2) \ast T(y_2, y_1)
Adding Inductive Definitions

\[ \text{Ring}_{h,t}() \leftarrow \exists y_1 \exists y_2 . \text{Chain}_{h,t}(y_1, y_2) * T(y_2, y_1) \]

\[ \text{Chain}_{h,t}(x, y) \leftarrow \exists z . S^{\text{token}}(x) * T(x, z) * \text{Chain}_{h,t+1}(z, y) \]

\[ \text{Chain}_{h,t}(x, y) \leftarrow \exists z . S^{\text{hole}}(x) * T(x, z) * \text{Chain}_{h-1,t+1}(z, y), \text{ where } n-1 \leq \max(0, n-1) \]
Adding Inductive Definitions

\[
\text{Ring}_{h,t}() \leftarrow \exists y_1 \exists y_2 . \text{Chain}_{h,t}(y_1, y_2) \cdot T(y_2, y_1)
\]

\[
\text{Chain}_{h,t}(x, y) \leftarrow \exists z . \text{S}_{\text{token}}(x) \cdot T(x, z) \cdot \text{Chain}_{h,t-1}(z, y)
\]

\[
\text{Chain}_{h,t}(x, y) \leftarrow \exists z . \text{S}_{\text{hole}}(x) \cdot T(x, z) \cdot \text{Chain}_{h-1,t}(z, y)
\]

\[
\text{Chain}_{0,1}(x, x) \leftarrow \text{S}_{\text{token}}(x)
\]

\[
\text{Chain}_{1,0}(x, x) \leftarrow \text{S}_{\text{hole}}(x)
\]

\[
\text{Chain}_{0,0}(x, x) \leftarrow \text{S}_{\text{token}}(x)
\]

\[
\text{Chain}_{0,0}(x, x) \leftarrow \text{S}_{\text{hole}}(x)
\]
Programmed Reconfigurability

- Sequential programming language based on:
  - primitives: `new(C,q,x)`, `delete(C,x)`, `connect(I,x₁, ..., xₙ)`, `disconnect(I,x₁, ..., xₙ)`
  - conditional: `with x₁, ..., xₙ : φ do ... od`, where φ is a CL formula with no predicates
  - sequential composition `(R₁; R₂)`, iteration `(R*)` and nondeterministic choice `(R₁ + R₂)`
An Example: Token Ring Node Removal
An Example: Token Ring Node Removal

with \( x, y, z : T(x, y) \cdot S_{\text{hole}}(y) \cdot T(y, z) \) do
\[ \text{disconnect}(T, x, y); \]
An Example: Token Ring Node Removal

with \( x, y, z : T(x, y) \ast S^{\text{hole}}(y) \ast T(y, z) \) do

\[ \text{disconnect}(T, x, y); \]
\[ \text{disconnect}(T, y, z); \]
An Example: Token Ring Node Removal

with \( x, y, z : T(x, y) \ast S^\text{hole}(y) \ast T(y, z) \) do
  disconnect(T, x, y);
  disconnect(T, y, z);
  delete(S, y);
An Example: Token Ring Node Removal

with x,y,z : T(x,y) * S_{hole}(y) * T(y,z) do
  disconnect(T,x,y);
  disconnect(T,y,z);
  delete(S,y);
  connect(x,z);
od
An Example: Token Ring Node Removal

with x,y,z : T(x,y) * S_{hole}(y) * T(y,z) do
  disconnect(T,x,y);
  disconnect(T,y,z);
  delete(S,y);
  connect(x,z);
od
An Example: Token Ring Node Removal

\[
\{ \text{Ring}_{2,1}() \} \\
\text{with } x,y,z : T(x,y) * S^\text{hole}(y) * T(y,z) \text{ do} \\
\quad \text{disconnect}(T,x,y); \\
\quad \text{disconnect}(T,y,z); \\
\quad \text{delete}(S,y); \\
\quad \text{connect}(x,z); \\
\text{od} \\
\{ \text{Ring}_{1,1}() \}
\]

We want to prove that, after the reconfiguration, there is still a token left.
Local Reasoning

```plaintext
{emp} new(C,q,x) {C^q(x)}
{C(x)} delete(C,x) {emp}
{emp} connect(I,x_1, ..., x_n) {l(x_1, ...., x_n)}
{l(x_1, ...., x_n)} disconnect(I,x_1, ..., x_n) {emp}
```

A local specification only mentions those resources that are necessary to avoid faulting
Local Reasoning

\[
\{\text{emp}\} \text{ new}(C,q,x) \{C^q(x)\}
\]
\[
\{C(x)\} \text{ delete}(C,x) \{\text{emp}\}
\]
\[
\{\text{emp}\} \text{ connect}(I,x_1,\ldots,x_n) \{l(x_1,\ldots,x_n)\}
\]
\[
\{l(x_1,\ldots,x_n)\} \text{ disconnect}(I,x_1,\ldots,x_n) \{\text{emp}\}
\]

A **local specification** only mentions those resources that are necessary to avoid faulting

\[
\{\phi\} \ R \ \{\psi\}
\]
\[
\{\phi \ast F\} \ R \ \{\psi \ast F\}
\]

if \( R \) is a **local program** and

\[
\text{modifies}(R) \cap \text{fv}(F) = \emptyset
\]

The **frame rule** plugs a local specification into a global context
Which Reconfiguration Programs are Local?

Let $\Gamma$ be the set of configurations

An action is a function $f : \Gamma \rightarrow \text{pow}(\Gamma)^T$, where $f(\gamma) = T \iff f$ faults in $\gamma$
Which Reconfiguration Programs are Local?

Let \( \Gamma \) be the set of configurations

An action is a function \( f : \Gamma \rightarrow \text{pow}(\Gamma)^T \), where \( f(\gamma) = T \Leftrightarrow f \) faults in \( \gamma \)

An action \( f \) is local \( \Leftrightarrow f(\gamma_1 \ast \gamma_2) \subseteq f(\gamma_1) \ast \{\gamma_2\} \), where \( S \subseteq T \), for each \( S \) in \( \text{pow}(\Gamma) \)

- new\((C,q,x)\), delete\((C,x)\), connect\((I,x_1, \ldots, x_n)\), disconnect\((I,x_1, \ldots, x_n)\)

- with \( x_1, \ldots, x_n : \phi \) do ... od, where \( \phi \) is a conjunction of equalities

- nondeterministic choices \( R_1 + R_2 \) between local programs
Which Reconfiguration Programs are Local?

Let $\Gamma$ be the set of configurations

An action is a function $f : \Gamma \rightarrow \text{pow}(\Gamma)^T$, where $f(\gamma) = T \Leftrightarrow f$ faults in $\gamma$

An action $f$ is local $\Leftrightarrow f(\gamma_1 * \gamma_2) \subseteq f(\gamma_1) * \{\gamma_2\}$, where $S \subseteq T$, for each $S$ in $\text{pow}(\Gamma)$

- $\text{new}(C,q,x)$, $\text{delete}(C,x)$, $\text{connect}(I,x_1, ..., x_n)$, $\text{disconnect}(I,x_1, ..., x_n)$
- with $x_1, ..., x_n : \phi$ do ... od, where $\phi$ is a conjunction of equalities
- nondeterministic choices $R_1 + R_2$ between local programs

Non-local programs:

- sequential compositions $R_1; R_2$
- with $x_1, ..., x_n : \phi$ do ... od, where $\phi$ contains component/interaction atoms
Non-local Rules

\[
\{\phi \land (\theta \ast \text{true})\} \Rightarrow \{\Psi\} \\
\{\forall x . \neg(\theta \ast \text{true}) \lor \phi\} \text{ with } x:\theta \text{ do } R \text{ od } \{\exists x . \Psi\}
\]
Non-local Rules

\[
\begin{align*}
\{ \phi \land (\theta \ast \text{true}) \} & \quad R \quad \{ \psi \} \\
\{ \forall x . \neg(\theta \ast \text{true}) \lor \phi \} & \quad \text{with } x:\theta \text{ do } R \text{ od } \{ \exists x . \psi \}
\end{align*}
\]

fv(\phi) \cap x = \emptyset

\[
\begin{align*}
\{ \phi \} & \quad R_1 \quad \{ \theta \} \\
\{ \theta \} & \quad R_2 \quad \{ \psi \}
\end{align*}
\]

\{ \phi \} \quad R_1; \quad R_2 \quad \{ \psi \}

\theta \text{ is havoc invariant}
Non-local Rules

\[
\begin{align*}
\{\phi \land (\theta \ast \text{true})\} & \ R \ \{\psi\} \\
\{\forall x . \neg(\theta \ast \text{true}) \lor \phi\} & \text{with } x:\theta \text{ do } R \text{ od } \{\exists x . \psi\} \\
\{\phi\} \ R_1 \ \{\theta\} & \quad \{\theta\} \ R_2 \ \{\psi\} \\
\{\phi\} \ R_1; \ R_2 \ \{\psi\} & \quad \theta \text{ is havoc invariant} \\
\{\phi\} \ R_1 \ \{\psi\} & \quad \{\phi\} \ R_2 \ \{\psi\} \\
\{\phi\} \ R_1 + \ R_2 \ \{\psi\} & \\
fv(\phi) \cap x = \emptyset
\end{align*}
\]
Non-local Rules

\[
\{\phi \land (\theta \ast \text{true})\} \quad R \quad \{\psi\}
\]

\[
\{\forall x . \neg(\theta \ast \text{true}) \lor \phi\} \quad \text{with} \quad x:\theta \quad \text{do} \quad R \quad \text{od} \quad \{\exists x . \psi\}
\]

\[
\{\phi\} \quad R_1 \quad \{\theta\} \quad \quad \{\theta\} \quad R_2 \quad \{\psi\}
\]

\[
\{\phi\} \quad R_1 + R_2 \quad \{\psi\}
\]

\[
\{\phi\} \quad R \quad \{\phi\}
\]

\[
\{\phi\} \quad R^* \quad \{\phi\}
\]

fv(\phi) \cap x = \emptyset

\(\theta\) is havoc invariant

\(\phi\) is havoc invariant
Havoc Invariance

A formula $\phi$ is **havoc invariant** $\iff$ for each model $\gamma$ of $\phi$ and each state change $\gamma \rightarrow \gamma'$ corresponding to firing one or more interactions enabled in $\gamma$, we have that $\gamma'$ is a model of $\phi$.

Havoc invariance is related to **entailment** $\phi \models \psi \iff$ every model of $\phi$ is a model of $\psi$. 
Structural Rules

\[
\begin{align*}
\{ \phi' \} & \mathcal{R} \{ \psi' \} & \text{if} & \quad \phi \vdash \phi' \\
\{ \phi \} & \mathcal{R} \{ \psi \} & & \\
\end{align*}
\]

\[
\begin{align*}
\{ \phi_i \} & \mathcal{R} \{ \psi_i \}, \quad i=1..k \\
\{ \bigvee_{i=1}^k \phi_i \} & \mathcal{R} \{ \bigvee_{i=1}^k \psi_i \} \\
\end{align*}
\]

\[
\begin{align*}
\{ \phi_i \} & \mathcal{R} \{ \psi_i \}, \quad i=1..k \\
\{ \bigwedge_{i=1}^k \phi_i \} & \mathcal{R} \{ \bigwedge_{i=1}^k \psi_i \} \\
\end{align*}
\]
Back to the Proof

\{\text{Ring}_{2,1}()\}

with \(x, y, z : T(x, y) \ast S_{\text{hole}}(y) \ast T(y, z)\) do

\hspace{1cm} \text{disconnect}(T, x, y);

\hspace{1cm} \text{disconnect}(T, y, z);

\hspace{1cm} \text{delete}(S, y);

\hspace{1cm} \text{connect}(x, z);

od

\{\text{Ring}_{1,1}()\}
Back to the Proof

{Ring$_{2,1}$()}
{∃a∃b . Chain$_{2,1}(a,b) * T(a,b)$}
{∀x∀y∀z . ¬[T(x,y) * S^{hole}(y) * T(y,z) * true] \lor ∃a∃b . Chain$_{2,1}(a,b) * T(a,b)$}
with x,y,z : T(x,y) * S^{hole}(y) * T(y,z) do

  disconnect(T,x,y);

  disconnect(T,y,z);

  delete(S,y);

  connect(x,z);

od

{∃a∃b . Chain$_{1,1}(a,b) * T(a,b)$}
{Ring$_{1,1}$()}


Back to the Proof

\{\text{Ring}_{2,1}()\}
\{\exists a \exists b . \text{Chain}_{2,1}(a,b) * T(a,b)\}
\{\forall x \forall y \forall z . \neg [T(x,y) * S^\text{hole}(y) * T(y,z) * \text{true}] \lor \exists a \exists b . \text{Chain}_{2,1}(a,b) * T(a,b)\}

with \(x, y, z : T(x,y) * S^\text{hole}(y) * T(y,z)\) do
\{\exists a \exists b . \text{Chain}_{2,1}(a,b) * T(a,b) \land [T(x,y) * S^\text{hole}(y) * T(y,z) * \text{true}]\}
\{T(x,y) * S^\text{hole}(y) * T(y,z) * \text{Chain}_{1,1}(z,x)\}

\text{disconnect}(T,x,y);
\text{disconnect}(T,y,z);
\text{delete}(S,y);
\text{connect}(x,z);
\{\text{Chain}_{1,1}(z,x) * T(x,z)\}
\text{od}
\{\exists a \exists b . \text{Chain}_{1,1}(a,b) * T(a,b)\}
\{\text{Ring}_{1,1}()\}
Back to the Proof

\{\text{Ring}_{2,1}()\}
\{\exists a\exists b . \text{Chain}_{2,1}(a,b) * T(a,b)\}
\{\forall x\forall y\forall z . \neg[T(x,y) * S_{\text{hole}}(y) * T(y,z) * \text{true}] \lor \exists a\exists b . \text{Chain}_{2,1}(a,b) * T(a,b)\}

with \(x,y,z : T(x,y) * S_{\text{hole}}(y) * T(y,z)\) do
\{\exists a\exists b . \text{Chain}_{2,1}(a,b) * T(a,b) \land [T(x,y) * S_{\text{hole}}(y) * T(y,z) * \text{true}]\}
\{T(x,y) * S_{\text{hole}}(y) * T(y,z) * \text{Chain}_{1,1}(z,x)\}

\begin{align*}
\text{disconnect}(T,x,y); \\
\text{disconnect}(T,y,z); \\
\text{delete}(S,y); \\
\text{connect}(x,z);
\end{align*}
\{\text{Chain}_{1,1}(z,x) * T(x,z)\}
\text{od}
\{\exists a\exists b . \text{Chain}_{1,1}(a,b) * T(a,b)\}
\{\text{Ring}_{1,1}()\}
Back to the Proof

\{Ring_{2,1}()\}
\{\exists a \exists b . \text{Chain}_{2,1}(a,b) * T(a,b)\}
\{\forall x \forall y \forall z . \neg [T(x,y) * S_{\text{hole}}(y) * T(y,z) * \text{true}] \lor \exists a \exists b . \text{Chain}_{2,1}(a,b) * T(a,b)\}

with \( x, y, z : T(x,y) * S_{\text{hole}}(y) * T(y,z) \) do
\{\exists a \exists b . \text{Chain}_{2,1}(a,b) * T(a,b) \land [T(x,y) * S_{\text{hole}}(y) * T(y,z) * \text{true}]\}
\{T(x,y) * S_{\text{hole}}(y) * T(y,z) * \text{Chain}_{1,1}(z,x)\}
  \text{ disconnect}(T,x,y); 
\{S_{\text{hole}}(y) * T(y,z) * \text{Chain}_{1,1}(z,x)\}
  \text{ disconnect}(T,y,z); 

\text{ delete}(S,y); 
\text{ connect}(x,z); 
\{\text{Chain}_{1,1}(z,x) * T(x,z)\}
\text{ od}\n\{\exists a \exists b . \text{Chain}_{1,1}(a,b) * T(a,b)\}
\{Ring_{1,1}()\}
{Ring\(_2,1()\)}
{\exists a \exists b \cdot \text{Chain}\(_2,1(a,b) \land T(a,b)\)}
{\forall x \forall y \forall z \cdot \neg[T(x,y) \land S^\text{hole}(y) \land T(y,z) \land \text{true}] \lor \exists a \exists b \cdot \text{Chain}\(_2,1(a,b) \land T(a,b)\)}

with \(x, y, z : T(x,y) \land S^\text{hole}(y) \land T(y,z)\) do
{\exists a \exists b \cdot \text{Chain}\(_2,1(a,b) \land T(a,b) \land [T(x,y) \land S^\text{hole}(y) \land T(y,z) \land \text{true}]]\}
{T(x,y) \land S^\text{hole}(y) \land T(y,z) \land \text{Chain}\(_1,1(z,x)\)}

\begin{align*}
\text{disconnect}(T,x,y); \\
S^\text{hole}(y) \land T(y,z) \land \text{Chain}\(_1,1(z,x)\) \\
\text{disconnect}(T,y,z);
\end{align*}

\begin{align*}
\text{delete}(S,y); \\
\text{connect}(x,z); \\
\text{Chain}\(_1,1(z,x) \land T(x,z)\)
\end{align*}

\text{od}
{\exists a \exists b \cdot \text{Chain}\(_1,1(a,b) \land T(a,b)\)}
{\text{Ring}\(_1,1()\)}
Back to the Proof

{\textbf{Ring}_{2,1}()}
{\exists a \exists b \ . \ \textbf{Chain}_{2,1}(a,b) \land T(a,b)}
{\forall x \forall y \forall z \ . \ \neg \left[ T(x,y) \land S^{\text{hole}}(y) \land T(y,z) \land \text{true} \right] \lor \exists a \exists b \ . \ \textbf{Chain}_{2,1}(a,b) \land T(a,b)}

with \( x, y, z : T(x,y) \land S^{\text{hole}}(y) \land T(y,z) \) do

{\exists a \exists b \ . \ \textbf{Chain}_{2,1}(a,b) \land T(a,b) \land \left[ T(x,y) \land S^{\text{hole}}(y) \land T(y,z) \land \text{true} \right]}

{\left[ T(x,y) \land S^{\text{hole}}(y) \land T(y,z) \land \text{Chain}_{1,1}(z,x) \right]}

\text{disconnect}(T,x,y);

{\left[ S^{\text{hole}}(y) \land T(y,z) \land \text{Chain}_{1,1}(z,x) \right]}

\text{disconnect}(T,y,z);

{\left[ S^{\text{hole}}(y) \land \text{Chain}_{1,1}(z,x) \right]}

\text{delete}(S,y);

\text{connect}(x,z);

{\left[ \text{Chain}_{1,1}(z,x) \land T(x,z) \right]}

od

{\exists a \exists b \ . \ \textbf{Chain}_{1,1}(a,b) \land T(a,b)}

{\textbf{Ring}_{1,1}()}
Back to the Proof

{Ring\(_2,1()\)}
{\exists a \exists b . \text{Chain}_2,1(a,b) * T(a,b)}
{\forall x \forall y \forall z . \neg [T(x,y) * S_{\text{hole}}(y) * T(y,z) * \text{true}] \lor \exists a \exists b . \text{Chain}_2,1(a,b) * T(a,b)}

with \(x,y,z: T(x,y) * S_{\text{hole}}(y) * T(y,z)\) do

{\exists a \exists b . \text{Chain}_2,1(a,b) * T(a,b) \land [T(x,y) * S_{\text{hole}}(y) * T(y,z) * \text{true}]}\)

{T(x,y) * S_{\text{hole}}(y) * T(y,z) * \text{Chain}_1,1(z,x)}
  disconnect(T,x,y);

{S_{\text{hole}}(y) * T(y,z) * \text{Chain}_1,1(z,x)}
  disconnect(T,y,z);

{S_{\text{hole}}(y) * \fbox{\text{Chain}_1,1(z,x)}}
  delete(S,y);

  connect(x,z);

{\text{Chain}_1,1(z,x) * T(x,z)}
od

{\exists a \exists b . \text{Chain}_1,1(a,b) * T(a,b)}
{\text{Ring}_1,1()}
Back to the Proof

{\text{Ring}_2,1()}
{\exists a \exists b . \text{Chain}_2,1(a,b) * T(a,b)}
{\forall x \forall y \forall z . \neg [T(x,y) * S_{\text{hole}}(y) * T(y,z) * \text{true}] \lor \exists a \exists b . \text{Chain}_2,1(a,b) * T(a,b)}

with x,y,z : T(x,y) * S_{\text{hole}}(y) * T(y,z) do
{\exists a \exists b . \text{Chain}_2,1(a,b) * T(a,b) \land [T(x,y) * S_{\text{hole}}(y) * T(y,z) * \text{true}]}
{\text{T}(x,y) * S_{\text{hole}}(y) * T(y,z) * \text{Chain}_{1,1}(z,x)}
  \text{disconnect}(T,x,y);
{S_{\text{hole}}(y) * T(y,z) * \text{Chain}_{1,1}(z,x)}
  \text{disconnect}(T,y,z);
{S_{\text{hole}}(y) * \text{Chain}_{1,1}(z,x)}
  \text{delete}(S,y);
{\text{Chain}_{1,1}(z,x)}
  \text{connect}(x,z);
{\text{Chain}_{1,1}(z,x) * T(x,z)}
\text{od}
{\exists a \exists b . \text{Chain}_{1,1}(a,b) * T(a,b)}
{\text{Ring}_1,1()}
Back to the Proof

{\text{Ring}_{2,1}()} \\
\{\exists a \exists b . \text{Chain}_{2,1}(a,b) \land T(a,b)\} \\
\{\forall x \forall y \forall z . \neg[T(x,y) \land S_{\text{hole}}(y) \land T(y,z) \land \text{true}] \lor \exists a \exists b . \text{Chain}_{2,1}(a,b) \land T(a,b)\} \\
\text{with } x,y,z : T(x,y) \land S_{\text{hole}}(y) \land T(y,z) \text{ do} \\
\{\exists a \exists b . \text{Chain}_{2,1}(a,b) \land T(a,b) \land [T(x,y) \land S_{\text{hole}}(y) \land T(y,z) \land \text{true}]\} \\
\{T(x,y) \land S_{\text{hole}}(y) \land T(y,z) \land \text{Chain}_{1,1}(z,x)\} \\
\text{disconnect}(T,x,y); \\
\{S_{\text{hole}}(y) \land T(y,z) \land \text{Chain}_{1,1}(z,x)\} \\
\text{disconnect}(T,y,z); \\
\{S_{\text{hole}}(y) \land \text{Chain}_{1,1}(z,x)\} \\
\text{delete}(S,y); \\
\{\text{Chain}_{1,1}(z,x)\} \\
\text{connect}(x,z); \\
\{\text{Chain}_{1,1}(z,x) \land T(x,z)\} \\
\text{od} \\
\{\exists a \exists b . \text{Chain}_{1,1}(a,b) \land T(a,b)\} \\
\{\text{Ring}_{1,1}()\}
Back to the Proof

\[
\{\text{Ring}_{2,1}()\} \\
\{\exists a \exists b . \text{Chain}_{2,1}(a,b) \land T(a,b)\} \\
\{\forall x \forall y \forall z . \neg [T(x,y) \land S^\text{hole}(y) \land T(y,z) \land \text{true}] \lor \exists a \exists b . \text{Chain}_{2,1}(a,b) \land T(a,b)\} \\
\text{with } x, y, z : T(x,y) \land S^\text{hole}(y) \land T(y,z) \text{ do} \\
\{\exists a \exists b . \text{Chain}_{2,1}(a,b) \land T(a,b) \land [T(x,y) \land S^\text{hole}(y) \land T(y,z) \land \text{true}]\} \\
\{T(x,y) \land S^\text{hole}(y) \land T(y,z) \land \text{Chain}_{1,1}(z,x)\} \\
\quad \text{disconnect}(T, x, y); \\
\{S^\text{hole}(y) \land T(y,z) \land \text{Chain}_{1,1}(z,x)\} \\
\quad \text{disconnect}(T, y, z); \\
\{S^\text{hole}(y) \land \text{Chain}_{1,1}(z,x)\} \\
\quad \text{delete}(S, y); \\
\{\text{Chain}_{1,1}(z,x)\} \\
\quad \text{connect}(x, z); \\
\{\text{Chain}_{1,1}(z,x) \land T(x, z)\} \\
\od \\
\{\exists a \exists b . \text{Chain}_{1,1}(a,b) \land T(a,b)\} \\
\{\text{Ring}_{1,1}()\}
\]
Proving Havoc Invariance

Axioms and inference rules for havoc triples of the form:

$$\eta \triangleright \{\phi\} \ L \ \{\Psi\}$$

where $\phi$ and $\Psi$ are CL formulae and $L$ is an extended regular expression:

- $\epsilon$ is the empty language
- $\Sigma[\alpha]$ is an alphabet symbol, $\alpha$ is an interaction or a predicate atom, and $\eta$ is a set of such symbols
- $L_1 \cdot L_2$ is the concatenation of $L_1$ and $L_2$
- $L_1 \cup L_2$ is the union of $L_1$ and $L_2$
- $L^*$ is the iteration of $L$
- $L_1 \bowtie_{\eta_1, \eta_2} L_2$ is the interleaving (zip) product of $L_1$ and $L_2$ w.r.t. the alphabets $\eta_1$ and $\eta_2$
Havoc Triples

\[ \models \eta \triangleright \{\{\phi\}\} \land \{\{\psi\}\} \]

for each each sequence \(\omega\) of interactions in the language of \(L\):

\(\gamma\) is a model of \(\phi\) and \(\gamma \xrightarrow{\omega} \gamma' \Rightarrow \gamma'\) is a model of \(\psi\)

NB: an interaction is executed in this context even with dangling ports

Proposition. \(\models \eta \triangleright \{\{\phi\}\} \Sigma[\phi]^* \{\{\psi\}\} \Rightarrow \phi\) is havoc invariant, where

\[\Sigma[\phi] \triangleq \bigcup\{\Sigma[\alpha] \mid \alpha\) is an interaction or a predicate atom of \(\phi\}\]
Regular Expression Rules

\[
\begin{align*}
\eta \triangleright \{\{\phi\}\} L_1 \{\{\theta\}\} & \quad \eta \triangleright \{\{\theta\}\} L_2 \{\{\psi\}\} \\
\eta \triangleright \{\{\phi\}\} L_1 \cdot L_2 \{\{\theta\}\} & \quad \eta \triangleright \{\{\phi\}\} L \{\{\phi\}\} \\
\eta \triangleright \{\{\phi\}\} L^* \{\{\phi\}\} & \quad \eta \triangleright \{\{\phi\}\} L_1 \{\{\psi\}\} \quad \quad \eta \triangleright \{\{\phi\}\} L_2 \{\{\psi\}\} \\
\eta \triangleright \{\{\phi\}\} L_1 \cup L_2 \{\{\psi\}\} & \quad \eta \triangleright \{\{\phi\}\} L_1 U L_2 \{\{\psi\}\} \\
\eta \triangleright \{\{\phi\}\} L_1 U L_2 \{\{\psi\}\} & \quad \eta \triangleright \{\{\phi\}\} L_1 \{\{\psi\}\}
\end{align*}
\]

\(\phi\) and \(\theta\) differ only by a renaming of states in the component atoms \(C^q(x)\).
The Composition Rule

\[ \eta_i \triangleright \{\phi_i \ast F(\phi_i, \phi_3^i)\} \text{ L}_i \{\psi_i \ast F(\phi_i, \phi_3^i)\}, \quad i = 1, 2 \]

\[ \eta_1 \cup \eta_2 \triangleright \{\phi_1 \ast \phi_2\} \text{ L}_1 \bowtie \eta_1, \eta_2 \text{ L}_2 \{\psi_1 \ast \psi_2\} \]

\[ \eta_i = \Sigma[\phi_i \ast F(\phi_i, \phi_3^i)] \]
The Composition Rule

\[ \eta_i \triangleright \{\phi_i \ast F(\phi_i, \phi_{3-i})\} \text{ L } \{\psi_i \ast F(\phi_i, \phi_{3-i})\}, \text{ i = 1,2} \]

\[ \eta_1 \cup \eta_2 \triangleright \{\phi_1 \ast \phi_2\} \text{ L } \eta_1 \bowtie \eta_1, \eta_2 \text{ L } \{\psi_1 \ast \psi_2\} \]

\[ \eta_i = \Sigma[\phi_i \ast F(\phi_i, \phi_{3-i})] \]
The Composition Rule

\[ \eta_i \triangleright \{\phi_i \ast F(\phi_i, \phi_{3-i})\} \quad L_i \{\psi_i \ast F(\phi_i, \phi_{3-i})\}, \quad i = 1, 2 \]

\[ \eta_1 \cup \eta_2 \triangleright \{\phi_1 \ast \phi_2\} \quad L_1 \bowtie \eta_1, \eta_2 \quad L_2 \{\psi_1 \ast \psi_2\} \]

\[ \eta_i = \Sigma[\phi_i \ast F(\phi_i, \phi_{3-i})] \]
The Composition Rule

\[ \eta_i \triangleright \{\phi_i \ast F(\phi_i, \phi_{3-i})\} L_i \{\psi_i \ast F(\phi_i, \phi_{3-i})\}, \ i = 1, 2 \]

\[ \eta_1 \cup \eta_2 \triangleright \{\phi_1 \ast \phi_2\} L_1 \bowtie_{\eta_1, \eta_2} L_2 \{\psi_1 \ast \psi_2\} \]

\[ \eta_i = \Sigma[\phi_i \ast F(\phi_i, \phi_{3-i})] \]
The Composition Rule

\[ \eta_i \triangleright \{\{\phi_i \ast F(\phi_i, \phi_{3-i})\}\} L_i \{\{\psi_i \ast F(\phi_i, \phi_{3-i})\}\}, \ i = 1, 2 \]

\[ \frac{\eta_1 \cup \eta_2 \triangleright \{\{\phi_1 \ast \phi_2\}\} \land_{\eta_1, \eta_2} L_1 \{\{\psi_1 \ast \psi_2\}\}}{\eta_i = \Sigma[\phi_i \ast F(\phi_i, \phi_{3-i})]} \]
The Composition Rule

\[ \eta_i \triangleright \{\phi_i \ast F(\phi_i, \phi_{3-i})\} \ L_i \{\psi_i \ast F(\phi_i, \phi_{3-i})\}, \ i = 1, 2 \]

\[ \eta_1 \cup \eta_2 \triangleright \{\phi_1 \ast \phi_2\} \ L_1 \bowtie_{\eta_1, \eta_2} L_2 \{\psi_1 \ast \psi_2\} \]

\[ \eta_i = \Sigma[\phi_i \ast F(\phi_i, \phi_{3-i})] \]
The Unfolding Rule

\[(\eta \setminus \Sigma[A(y_1, \ldots, y_n)]) \cup \Sigma[\rho'] \triangleright \{\phi^* \rho'\} \quad L[\Sigma[A(y_1, \ldots, y_n)] / \Sigma[\rho']] \quad \{\psi\}\]

\[\eta \triangleright \{\phi^* A(y_1, \ldots, y_n)\} \quad L \quad \{\psi\}\]

\[A(x_1, \ldots, x_n) \leftarrow \rho \text{ is a rule defining } A\]

\[\rho' = \rho[x_1/y_1, \ldots, x_n/y_n]\]
The Unfolding Rule

\[(\eta \setminus \Sigma[A(y_1, \ldots, y_n)]) \cup \Sigma[\rho'] \triangleright \{(\phi^* \rho')\} \ \text{L}[\Sigma[A(y_1, \ldots, y_n)] / \Sigma[\rho']] \ {\{\psi\}\}}\]

\[\eta \triangleright \{(\phi^* A(y_1, \ldots, y_n))\} \ \text{L} \ {\{\psi\}\} \]

\[A(x_1, \ldots, x_n) \leftarrow \rho \text{ is a rule defining } A\]
\[\rho' = \rho[x_1/y_1, \ldots, x_n/y_n] \]

- each model of \(A(y_1, \ldots, y_n)\) is obtained from a finite number of unfoldings, called a \textit{stage number}\n
- the stage number multiset of the submodels of \(\phi^* A(y_1, \ldots, y_n)\) corresponding to predicate atoms strictly decreases in the \textit{multiset order} [Dershowitz & Manna 1979]
Cyclic Proofs

There is no right triangle whose sides are integers whose area is equal to the square of an integer. If there is a triangle ... there is a second triangle, smaller than the first, which has the same property. And if there is a second, smaller than the first ... then there is, by like reasoning, a third smaller than the second .... and so on ad infinitum. But there is not an infinite number of [positive] integers less than a given integer. [Fermat, 1659]
There is no right triangle whose sides are integers whose area is equal to the square of an integer. If there is a triangle there is a second triangle, smaller than the first, which has the same property. And if there is a second, smaller than the first then there is, by like reasoning, a third smaller than the second ... and so on ad infinitum. But there is not an infinite number of [positive] integers less than a given integer. [Fermat, 1659]
There is no right triangle whose sides are integers whose area is equal to the square of an integer. If there is a triangle ... there is a second triangle, smaller than the first, which has the same property. And if there is a second, smaller than the first ... then there is, by like reasoning, a third smaller than the second .... and so on ad infinitum. But there is not an infinite number of [positive] integers less than a given integer. [Fermat, 1659]
There is no right triangle whose sides are integers whose area is equal to the square of an integer. If there is a triangle ... there is a second triangle, smaller than the first, which has the same property. And if there is a second, smaller than the first ... then there is, by like reasoning, a third smaller than the second .... and so on ad infinitum. But there is not an infinite number of [positive] integers less than a given integer. [Fermat, 1659]
Cyclic Proofs

There is no right triangle whose sides are integers whose area is equal to the square of an integer. If there is a triangle ... there is a second triangle, smaller than the first, which has the same property. And if there is a second, smaller than the first ... then there is, by like reasoning, a third smaller than the second .... and so on ad infinitum. But there is not an infinite number of [positive] integers less than a given integer. [Fermat, 1659]
Cyclic Proofs

There is no right triangle whose sides are integers whose area is equal to the square of an integer. If there is a triangle ... there is a second triangle, smaller than the first, which has the same property. And if there is a second, smaller than the first ... then there is, by like reasoning, a third smaller than the second .... and so on ad infinitum. But there is not an infinite number of [positive] integers less than a given integer. [Fermat, 1659]
A Recap

- A simplified model of dynamic reconfigurable systems
  - components with finite-state behavior and interactions of finite arity
  - a sequential programming language for describing reconfiguration
- A resource logic for describing possibly infinite sets of configurations
  - inductively defined predicate symbols
- A proof system for reconfiguration programs
  - uses local reasoning to a maximum extent
  - generates external proof obligations (entailments and havoc invariants)
- A cyclic proof system for establishing havoc invariants
Future Work: Proof Automation

- Checking entailments automatically
  - undecidable problem
  - decidability requires several syntactic and a semantic restriction(s)
  - the restricted fragment was found to be 2EXPTIME-complete
  - semantic restriction of bounded degree excludes star architectures

- Checking havoc invariance automatically (work in progress)
  - undecidable problem
  - decidable for classes with decidable entailment

- Checking safety of a set of configurations described by a CL formula
  - undecidable problem
  - sound but incomplete method based on structural invariant synthesis