Safety and Liveness Properties in Copied Computational Artefacts

Nicola Angius \(^1\)  Giuseppe Primiero \(^2\)

\(^1\)Department of History, Human Sciences, and Education. University of Sassari, Italy
nangius@uniss.it

\(^2\)Department of Philosophy
University of Milan, Italy
giuseppe.primiero@unimi.it

Bertinoro, October 1, 2019
Motivations: Philosophical concerns

Ontology is essentially based on our understanding of identity

"No Entity without Identity", Quine (1969)

"No Entity without Type. No Type without Identity.", (PML, 1993)
Motivations: Philosophical concerns

Modularity and multiple realizability of computational systems requires the representation of identity, as well as weaker relations.

“No Program without Specification.
No Specification without Copies.”
Motivations: a trilogy

1. A taxonomy of relations for exact, inexact and approximate copies of computational artifacts – Angius and Primiero (2018)


3. Application to security: how to preserve higher order properties across copies – this work
Motivations

- **Ontology**: Are safety or liveness properties preserved by artefact copies?

- **Logic**: Can verification of safety and liveness properties be simplified for copied artefacts?

- **Ethics**: Should copying be allowed in the software industry if preserving safety or liveness properties?
Summary

- Recall the taxonomy of copy relations in Angius, Primiero (2018) for computational artefacts and their logical representations.

- Define safety and liveness properties of computational artefacts.

- Examine whether exact, inexact, and approximate copies preserve safety or liveness properties.

- Provide a model checking algorithm for inexact and approximate copy not preserving safety or liveness properties.

- Provide contraction and expansion algorithmic operations enabling to develop a safety- and liveness-preserving approximate copies.
Taxonomy of the copy relation, Angius and Primiero (2018)

<table>
<thead>
<tr>
<th>IDENTITY</th>
<th>EXACT COPY</th>
<th>INEXACT COPY</th>
<th>APPROXIMATE COPY</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S(P) = S(P)$</td>
<td>$S(P) = S(P)$</td>
<td>$S(P) \subset S(P')'$</td>
<td>$S(P) \cap S(P')'$</td>
</tr>
<tr>
<td>$L(P) = L(P)$</td>
<td>$L(P) = L(P)$</td>
<td>$L(P) = L(P')$</td>
<td>$L(P) = L(P')$</td>
</tr>
<tr>
<td>$I(P) = I(P)$</td>
<td>$I(P) \neq I(P')$</td>
<td>$I(P) \neq I(P')'$</td>
<td>$I(P) \neq I(P')$</td>
</tr>
</tbody>
</table>
Formal Representation of Computational Systems

Definition (Finite State Transition System)
A finite transition system

\[ TS = (S, A, T, I, F, AP, L) \]

is a set theoretic structure where:

- \( S = (s_0, \ldots, s_n) \) is a finite set of states;
- \( A \) is a set of finite transitions’ labels;
- \( T \subseteq S \times A \times S \) is a transition relation;
- \( I \subseteq S \) is a set of initial states;
- \( F \subseteq S \) is a set of final states;
- \( AP \) is a finite set of states’ labels;
- \( L : S \rightarrow 2^{AP} \) is the function labelling states.
A behaviour for such a system is a sequence of functionalities, each satisfied at a state and which are realised by a path:

**Definition (Path)**

Given a finite transition system $TS$, a finite path fragment is a finite sequence of states $\pi = (s_0, \ldots, s_n)$, such that each $s_i$ is the successor of $s_{i-1}$. An infinite path fragment is an infinite sequence of states $\pi = (s_0, s_1, \ldots)$ such that each $s_i$ is the successor of $s_{i-1}$. A path is a path fragment which starts in an initial state $s_i \in I$ and either terminates in a final state $s_j \in F$, or is infinite.
Satisfiability

Definition (Satisfaction of CTL* formulas)

\(TS, s_i \models p\) iff \(p \in L(s_i)\);
\(TS, s_i \models \neg p\) iff \(TS, s_i \not\models p\);
\(TS, s_i \models p \land q\) iff \(TS, s_i \models p\) and \(TS, s_i \models q\);
\(TS, s_i \models Xp\) iff there exists \(s_j \in S\) s.t. \(s_i Ts_j\) and \(TS, s_j \models p\);
\(TS, s_i \models Fp\) iff for all paths \(\pi = (s_i, s_j, \ldots)\) such that
there exists \(s_k \in \pi, TS, s_k \models p\);
\(TS, s_i \models Gp\) iff there exists a path \(\pi = (s_i, s_j, \ldots)\) such that
for all \(s_k \in \pi, TS, s_k \models p\);
\(TS, s_i \models [pUq]\) iff there exists a path \(\pi = (s_i, s_j, \ldots)\) and an index \(k\) such that
\(TS, s_k \models q\) and \(TS, s_j \models p\) for all \(j \leq k\);
\(TS, s_i \models \forall p\) iff for all paths \(\pi = (s_i, s_j, \ldots), TS, \pi \models p\);
\(TS, s_i \models \exists p\) iff there exist a path \(\pi = (s_i, s_j, \ldots)\) such that \(TS, \pi \models p\);
\(TS, \pi \models Xp\) iff \(TS, \pi_1 \models p\);
\(TS, \pi \models Fp\) iff there exists \(k \geq 0\) such that \(TS, \pi_k \models p\);
\(TS, \pi \models Gp\) iff for all \(k \geq 0, TS, \pi_k \models p\);
\(TS, \pi \models [pUq]\) iff there exists \(k \geq 0\) such that \(TS, \pi_k \models q\) and \(TS, \pi_j \models p\) for all \(j \leq k\);
Bisimulation

Definition (Bisimulation)

Given two labelled transition systems $TS$ and $TS'$ on a finite set of states $S$, a bisimulation between them denoted as $TS \equiv TS'$ occurs when:

1. for each initial state $s_0$ in $TS$ there is an initial state $s'_0$ in $TS'$ that establishes a bisimulation relation $\equiv (s_0, s'_0)$, and vice versa;

2. $\equiv (s_0, s'_0)$ holds if and only if $s_0$ and $s'_0$ have the same label and there is a successor state $s'_1$ of $s'_0$ in $TS'$ for each successor state $s_1$ of $s_0$ in $TS$ such that $\equiv (s_1, s'_1)$. 
Simulation

Definition (Simulation)

Given two labelled transition systems $TS$ and $TS'$ on a finite set of states, $TS'$ is said to simulate $TS$ denoted by $TS \leq TS'$ when

1. for every initial state $s_0$ in $TS$ there is an initial state $s'_0$ in $TS'$ such that $(s_0, s'_0)$ holds;

2. $(s_0, s'_0)$ holds if $s_0$ and $s'_0$ have the same label and there is a successor state $s'_1$ of $s'_0$ in $TS'$ for each successor state $s_1$ of $s_0$ in $TS$ such that $(s_1, s'_1)$. 
Theorem (Exact Copy as Bisimulation)

TS' is an exact copy of TS if and only if \( TS \equiv TS' \).
Inexact copy

Theorem (Inexact Copy as Simulation)

TS' is an inexact copy of TS if and only if TS ≤ TS'.

```
start → 0
  ^   ^
a   a
  |   |
1   1
  ↓   ↓
b   c
  |   |
2   3
```
Theorem (Approximate copy)

TS' is an approximate copy of TS iff \( \exists \pi' \in TS' / \leq s.t. \pi' \leq TS \).
Safety properties

- **Safety**: something “bad” will never happen.

- Example for a traffic lights control system: “it will not ever happen that all traffic lights are green simultaneously”.

- Formalization in $CTL^*$: $\text{G} \neg (green_1 \land \cdots \land green_n)$
Liveness properties

- **Liveness**: something “good” will always happen.

- Example for a traffic lights control system: “each traffic light will eventually turn green infinitely often”.

- Formalization in $CTL^*$: $\text{GF}(\bigvee \cdots \bigvee green_n)$
Corollary (Safety and liveness in exact copies)

Exact copies preserve both safety and liveness properties.

Proof.

Theorem (Bisimulation and $CTL^*$ equivalence)

$TS \equiv TS'$ iff $TS \equiv_{CTL^*} TS'$, i.e. $TS, s_i \models g \leftrightarrow TS', s_i' \models g$, for any $CTL^*$ formula $g$. 

$\square$
Corollary (Safety and liveness in inexact copy)

\[ TS \preceq TS' \text{ iff } TS', s_i' \vDash \psi \rightarrow TS, s_i \vDash \psi, \text{ for any safety or liveness property } \psi \]

Proof.

Theorem (Simulation and \( \forall CTL^* \) equivalence)

\[ TS \preceq TS' \text{ iff } TS', s_i' \vDash g \rightarrow TS, s_i \vDash g, \text{ for any } \forall CTL^* \text{ formula } g. \]

\( \square \)
Corollary (Violation of safety and liveness in inexact copy)
\[ TS \leq TS' \text{ iff } TS, s_i \models \neg \psi \rightarrow TS', s'_i \models \neg \psi, \text{ for any safety or liveness property } \psi. \]

Proof.

Theorem (Simulation and \(\exists\)CTL\(^*\) equivalence)
\[ TS \leq TS' \text{ iff } TS, s_i \models g \rightarrow TS', s'_i \models g, \text{ for any } \exists\text{CTL}^* \text{ formula } g. \]

In case \(TS \models \psi\), \(TS'\) may or may not satisfy \(\psi\), so \(TS'\) needs to be model checked against \(\psi\).
Verifying safety properties of inexact copies

Begin

Data: $TS' \oplus TS$, $A$, and $\chi$

Result: true if $TS' \models \psi$, false otherwise with a counterexample for $\psi$

initialization;

$R := \{(s_i, s_j) \in TS' \oplus TS | L(s_i) = L(s_j)\}$;

$b := true$;

$\Pi := \emptyset$;
Verifying safety properties of inexact copies

while \( R \) is not a simulation do
  for \( s_i, s_j \in R; t, u \in T' \cup T \) do
    \((s_i, s_j) := (s_0', s_0)\);
    \((t, u) := (\alpha_0', \alpha_0)\);
    while \( \Pi = \emptyset \) do
      let \( s'_i \in Post(s_i, t), s'_j \in Post(s_j, u) \);
      if \( (s'_i, s'_j) \in R \) then
        \((s_i, s_j) := (s'_i, s'_j)\);
        \( t := \beta_1 \);
        \( u := \beta_2 \);
      else
        \( R := R \setminus \{(s_i, s_j)\} \);
        \( \Pi := \Pi \cup \{(s_0', \ldots, s'_i)\} \);
      end
    end
  end
end
Verifying safety properties of inexact copies

for $\pi \in \Pi \times L(A)$ do
  for $s_i \in \pi$ do
    if $s_i \not\models \chi$ then
      $b := \neg b$;
      $T := s_0, \ldots, s_i$
    end
  end
if $b$ then
  return("yes");
else
  return("no", $T$);
end
end
Algorithm properties

**Theorem (Partial Correctness of Algorithm 1)**

*On termination, Algorithm 1 returns the answer ’false’ if and only if $TS' \not\models \psi$.*

**Theorem (Termination of Algorithm 1)**

*Algorithm 1 terminates for $TS \leq TS'$.*

**Theorem (Time complexity of Algorithm 1)**

*The worst time complexity of Algorithm 1 is in $O((|R| \cdot |L(A)|) \cdot (1 + |\chi|) + M)$, where $M = \{|T' \cup T| \cdot |\delta|\}$.*
Verifying liveness properties of inexact copies

Begin

Data: $TS' \oplus TS, A,$ and $\nu$

Result: true if $TS' \models \phi$, false otherwise with a counterexample for $\phi$

initialization;

$\mathcal{R} := \{(s_i, s_j) \in TS' \oplus TS | L(s_i) = L(s_j)\}$;

$b := true$;

$\Pi := \emptyset$;
Verifying liveness properties of inexact copies

\[
\textbf{while } \mathcal{R} \text{ is not a simulation do}
\]
\[
\textbf{for } s_i, s_j \in \mathcal{I}; \ t, u, \in T' \cup T \textbf{ do}
\]
\[
(s_i, s_j) := (s'_0, s_0);
(t, u) := (\alpha'_0, \alpha_0);
\]
\[
\textbf{while } \Pi = \emptyset \textbf{ do}
\]
\[
\textbf{let } s'_i \in \text{Post}(s_i, t), s'_j \in \text{Post}(s_j, u);
\]
\[
\textbf{if } (s'_i, s'_j) \in \mathcal{R} \textbf{ then}
\]
\[
(s_i, s_j) := (s'_i, s'_j);
\]
\[
t := \beta_1;
\]
\[
u := \beta_2;
\]
\[
\textbf{else}
\]
\[
\mathcal{R} := \mathcal{R} \setminus \{(s_i, s_j)\};
\]
\[
\Pi := \Pi \cup \{(s'_0, \ldots, s'_i)\};
\]
\[
\textbf{end}
\]
\[
\textbf{end}
\]
\[
\textbf{end}
\]
for $\pi \in \Pi \times L(A)$ do
  for $s_i \in \pi$ do
    if $s_i \not\models \nu$ then
      $t := s_i$;
      if $s_i \in \text{Post}(t)$ then
        $b := \neg b$;
        $T := s_0, \ldots, s_i$;
      else
        for $s_j \in \text{Post}(t)$ do
          $t := s_j$;
          if $s_i \in \text{Post}(t)$ then
            $b := \neg b$;
            $T := s_0, \ldots, s_i, \ldots, s_i$;
          end
        end
      end
    end
  end
end
Verifying liveness properties of inexact copies

```plaintext
if b then
    return("yes");
else
    return("no", T);
end
end
```
Properties of Algorithm 2

Theorem (Partial Correctness of Algorithm 2)
On termination, Algorithm 2 returns the answer ‘false’ if and only if $TS' \not\models \phi$.

Theorem (Termination of Algorithm 2)
Algorithm 2 terminates for $TS \leq TS'$.

Theorem (Time complexity of Algorithm 2)
The worst time complexity of Algorithm 2 is in $O((|R| \cdot |L(A)|) \cdot (1 + |\nu|) + M)$, where $M = \{|T' \cup T| \cdot |\delta|\}$.
Safety and Liveness in approximate copies

Let us introduce $P = \{\pi' \in TS' / \leq |\pi' \leq TS\}$, then:

Corollary (Safety and liveness in approximate copies)
$P \leq TS$ iff $TS \models \psi \rightarrow P \models \psi$ for any safety or liveness property $\psi$.

Corollary (Violation of Safety and approximate copies)
If $P \leq TS$ then $TS \not\models \psi \rightarrow TS' \not\models \psi$ for any safety or liveness property $\psi$.

Proof.
If $TS \not\models \psi$, then also $P \not\models \psi$ because $P \in Path(TS)$; since also $P \in Path(TS')$ and $\psi$ is a $\forall CTL^*$ formula, it holds that $TS' \not\models \psi$. □

In case $TS \models \psi$, $TS'$ may or may not satisfy $\psi$, so $TS'$ needs to be model checked against $\psi$. 
Verifying safety and liveness of approximate copies

- **Safety**: only those paths of $TS'$ which are not in $P$ have to be checked against $\psi$. This can be done by adapting algorithm 1 which takes now inputs $TS' \oplus P$, $A$, and $\chi$ and returns true if $TS' \models \psi$ and false, together with a counterexample, if $TS' \not\models \psi$.

- **Liveness**: Algorithm 2 is adapted by taking $TS' \oplus P$, $A$, and $\nu$ as inputs and returning true if $TS' \models \phi$, false, together with a counterexample, if $TS' \not\models \phi$. 
Model and Priority

Definition

\[ S_i := \{ \Phi \cup \Psi \cup F \} \]

\( x_i < x_j \) for any two properties of \( S_i \) iff \( \exists \pi_i, \pi_j \in TS \) such that respectively \( s_i \in \pi_i, s_i \models x_i \) and \( s_j \in \pi_j, s_j \models x_j \) and \( s_i \rightarrow^a s_j \).
Contracted System

Definition
Given a transition system $TS = (S, A, T, I, F, AP, L)$ for $S_i$, a transition system $TS^{\circ} = (S^{\circ}, A^{\circ}, T^{\circ}, I^{\circ}, F^{\circ}, AP^{\circ}, L^{\circ})$ for $(S_i)_{i=1}^k$ is defined as follows:

- $S^{\circ} = S \setminus \bigcup_{i=0}^k \Xi_i$;
- $A^{\circ} = A$;
- $T^{\circ} \subseteq S^{\circ} \times A \times S^{\circ}$;
- $I^{\circ} \subseteq I$;
- $F^{\circ} \subseteq S^{\circ}$;
- $AP^{\circ} = AP$;
- $L^{\circ}: S^{\circ} \rightarrow 2^{AP^{\circ}}$. 


**Data:** $TS$

**Result:** $TS^\circ$

 initialization;

$TS := \Phi, \Psi, F;$

$F := \{f_1 < \cdots < f_n\};$

$\Psi := \{\psi_1 < \cdots < \psi_n\};$

$\Phi := \{\phi_1 < \cdots < \phi_n\};$

$\Xi := \emptyset;$

$f_i := f_n;$

$\psi_i := \psi_n;$

$\phi_i := \phi_n;$
for $f_i \in F, 1 \leq i < n, i := i + 1$ do
  for $\pi_i \in TS$ s.t.
    $\exists s_i \in \pi_i, s_i \models f_i, \psi_i \in \Psi, \phi_i \in \Phi, 1 \leq j < n, j := j + 1$ do
      $\Xi_i := \Xi_0 \cup s_i$;
      $TS := \{TS \setminus \Xi_i\}$;
      if $\forall \pi_j \in TS, \pi_j \models \psi_i \land \phi_i$;
        then
          $TS^\circ := \{TS\}$;
        else
          $TS^\circ := \{\text{Path}(TS) \cup \pi_i\}$;
        end
    end
  end
  $\psi_i := \psi_{n-j}$;
  $\phi_i := \phi_{n-j}$;
end

$f_i := f_{n-i}$;

$F := \{F \setminus \Xi_i\}$;

return $TS^\circ$
Expanded System

Definition
A transition system $TS' = (S', A', T', I', F', AP', L')$ for $(S_i)^+_{f_i}$ is defined as follows:

- $S' = S^\circ \cup S(f_i)$;
- $A' \supset A^\circ$;
- $T' \subseteq S' \times A' \times S'$;
- $I' \subseteq S'$;
- $F' \subseteq S'$;
- $AP' \supset AP^\circ$;
- $L' : S' \rightarrow 2^{AP'}$. 
Data: $TS^\circ, P$

Result: $TS^+$

initialization;

$TS^\circ := \Phi, \Psi, F^\circ$;

$s_i := s \in S \text{ s.t. } s \models g_i$;

$g_i := g_1$;

$\chi_i := \chi_n$;

$\nu_i := \nu_n$;
for $g_i \in G, 1 \leq i < n, i := i + 1$ do
  for $\chi_i, \upsilon_i, 1 \leq j < n, j := j + 1$ do
    if $s_i \models \chi_i$;
    then
      $S^+ := \{S^o \cup s_i\}$;
      for $f_j > g_i \in P$ do
        $T^+ := T \cup \{(s_k, s_i), (s_i, s_j), (s_j, s_l), (s_l, s_k) \mid s_j \in \Xi, s_k \in TS \setminus \Xi, s_l \models \upsilon\}$;
      end
    end
    $\chi_i := \chi_{n-j}$;
    $\upsilon_i := \upsilon_{n-j}$;
  end
  $g_i := g_{i+1}$;
end
$F := \{F \setminus \Xi_i\}$;
return $TS^+$
Conclusions

▶ Inexact and Approximate copies do not necessarily preserve safety and liveness properties.

▶ It is possible to improve model checking algorithms for these properties working with copies.

▶ Software development process can preserve safety and liveness properties through contraction and expansion algorithmic operations.
References

