Auxiliary variables and general specifications

Carsten Varming
Carnegie Mellon University

Cambridge Concurrency Workshop 2010
Auxiliary variables in CSL

Auxiliary variables track the effect of a process on the shared state.
Common uses:

- The auxiliary variables of a process are independent of the auxiliary variables in the other processes, reflecting, a process’ effect on a resource being independent of the other processes.

- The auxiliary variables of the processes stand in a relationship, thus restricting the interaction between processes.
Auxiliary variables in CSL

Example of independent auxiliary variables:

\[
\text{with } r \text{ do } x := x + 1
\]

With the invariant:

\[
r(x) : \{x = N + a\}
\]

we can derive:

\[
\{a = A\} \text{ with } r \text{ do } (a := a + 1; x := x + 1) \{a = A + 1\}
\]
We can easily extend with another process:
By substitution of $N + b$ for $N$:

\[ r(x) : \{ x = N + b + a \} \vdash \{ a = A \} \text{with } r \text{ do } (a := a + 1; x := x + 1) \{ a = A + 1 \} \]

thus with $r(x) : \{ x = N + b + a \} \vdash \{ b = B \} \text{with } r \text{ do } (b := b + 2; x := x + 2) \{ b = B + 2 \}$

we may conclude:

\[ r(x) : \{ x = N + b + a \} \vdash \{ a = A \land b = B \} \]

with $r$ do $(a := a + 1; x := x + 1) \parallel$

with $r$ do $(b := b + 2; x := x + 2)$

\[ \{ a = A + 1 \land b = B + 2 \} \]
Auxiliary variables in CSL

Example of processes with constrained interaction: Let

\[ \text{PUT}(x, a) \equiv \text{with } r \text{ when } full = 0 \text{ do } (full := 1; z := x; a := a + 1) \]

\[ \text{GET}(y, b) \equiv \text{with } r \text{ when } full = 1 \text{ do } (full := 0; y := z; b := b + 1) \]

With the invariant:

\[ r(z, full) : \{(full = 0 \land a = b) \lor (full = 1 \land z = X \land a = b + 1)\} \]

we can derive:

\[ \{a = A \land x = X\}\text{PUT}(x, a)\{a = A + 1\} \]

\[ \{b = B\}\text{GET}(y, b)\{b = B + 1 \land y = X\} \]
Auxiliary variables in CSL

Easy to show:

\[ \{ a = 0 \land b = 0 \land x = X \} \text{PUT}(x, a) \parallel \text{GET}(y, b) \]
\[ \{ a = 1 \land b = 1 \land y = X \} \]

but what about:

\[ (\text{PUT}(x_0, a); \text{PUT}(x_1, a)) \parallel (\text{GET}(y_0, b); \text{GET}(y_1, b)) \]
Auxiliary variables in CSL

With the invariant:

\[ r(z, \text{full}) : \{(\text{full} = 0 \land a = b) \lor (\text{full} = 1 \land z = X_b \land a = b + 1)\} \]

we can derive:

\[ \{a = 0 \land b = 0 \land x_0 = X_0 \land x_1 = X_1\} \]

\[ (\text{PUT}(x_0, a); \text{PUT}(x_1, a)) \parallel (\text{GET}(y_0, b); \text{GET}(y_1, b)) \]

\[ \{a = 2 \land b = 2 \land y_0 = X_0 \land y_1 = X_1\} \]
Auxiliary variables in CSL

With the invariant:

\[
\begin{align*}
r(z, full) & : \{(full = 0 \land a = b) \lor \\
& \quad (full = 1 \land z = X_b \land a = b + 1)\}\end{align*}
\]

we can derive:

\[
\begin{align*}
\{a = 0 \land b = 0 \land x_0 = X_0 \land x_1 = X_1\} \\
(PUT(x_0, a); PUT(x_1, a)) \parallel (GET(y_0, b); GET(y_1, b)) \\
\{a = 2 \land b = 2 \land y_0 = X_0 \land y_1 = X_1\}
\end{align*}
\]

\[
\begin{align*}
r(z, full) & : \{(full = 0 \land a = b) \lor \\
& \quad (full = 1 \land z = X \land a = b + 1)\}\end{align*}
\]
Higher Order Variables

With the invariant:

\[ F : \text{int} \rightarrow \text{int} \]

\[ r(z, full) : \{(full = 0 \land a = b) \lor (full = 1 \land z = F(b) \land a = b + 1)\} \]

we can derive:

\[ \{a = N \land x = F(N)\}\text{PUT}(x, a)\{a = N + 1\} \]

\[ \{b = N'\}\text{GET}(y, b)\{b = N' + 1 \land y = F(N')\} \]
Auxiliary variables in CSL

\[ \{ x_0 = N_0 \land x_1 = N_1 \land x_2 = N_2 \} \]

\[(\text{PUT}(x_0, a); \text{PUT}(x_1, a); \text{PUT}(x_2, a)) \parallel \]

\[\text{GET}(y_0, b) \parallel \text{GET}(y_1, c) \]

\[\{ y_0 \in \{N_0, N_1\} \land y_1 \in \{N_0, N_1\} \} \]
Auxiliary variables in CSL

\{x_0 = N_0 \land x_1 = N_1 \land x_2 = N_2\}

(PUT(x_0, a); PUT(x_1, a); PUT(x_2, a)) \parallel

GET(y_0, b) \parallel GET(y_1, c)

\{y_0 \in \{N_0, N_1\} \land y_1 \in \{N_0, N_1\}\}

Potential invariant:

\[ r(z, full) : \{(full = 0 \land a = M + b) \lor \]
\[ (full = 1 \land z = F(a - 1) \land a = M + b + 1)\} \]
New approach

Instead of invariants, for each resource we specify:

- The assumption on the state of the resource when entering a critical region.
- The effect of that critical region on the resource.
New approach

Instead of invariants, for each resource we specify:

- The assumption on the state of the resource when entering a critical region.
- The effect of that critical region on the resource.

The effect of each critical region is specified by a partial relation: $R \subseteq \text{states} \times \text{states}_\perp$.

The effect of a program is specified by a set of sequences of such partial relations.
Example

\[ \text{PUT}(x) \equiv \text{with } r \text{ when } full = 0 \text{ do } (full := 1; z := x) \]

If

\[
R_p = (\{\text{full} = 0\} \times \{\text{full} = 1 \land z = X\}) \cup \\
(\{\text{full} = 1\} \times \bot)
\]

then PUT can be given a specification \( r(\text{full}, z) : R_p \) so that
we can derive:

\[
\{x = X\} \text{PUT}(x) \{\text{true}\}
\]
Example

\[ \text{GET}(y) \equiv \text{with } r \text{ when full} = 1 \text{ do } (\text{full} := 0; \ y := z) \]

If

\[ R_g = (\{\text{full} = 1 \land z \in S\} \times \{\text{full} = 0\}) \cup \]
\[ (\{\text{full} = 0\} \times \bot) \]

then GET can be given a specification \( r(\text{full}, z) : R_g \) so that we can derive:

\[ \{\text{true}\} \text{GET}(y) \{y \in S\} \]
Example

We have

\(\{x_0 = N_0 \land x_1 = N_1 \land x_2 = N_2\}\)

\(\text{PUT}(x_0); \text{PUT}(x_1); \text{PUT}(x_2)\{\text{true}\}\)

with

\(r(\text{full}, z) : [N_0/X]R_p; [N_1/X]R_p; [N_2/X]R_p\)

which abbreviates:

\(r(\text{full}, z) : \langle [N_0/X]R_p, [N_1/X]R_p, [N_2/X]R_p \rangle\)
We have

\[
\{\text{true}\}(\text{GET}(y_0); \text{GET}(y_1)) \parallel \text{GET}(y_2)
\]

\[
\{y_0 \in \{N_0, N_1\} \land y_1 \in \{N_1, N_2\} \land y_2 \in \{N_0, N_1, N_2\}\}
\]

with

\[
r(\text{full}, z) : ([\{N_0, N_1\}/S]R_g; [\{N_1, N_2\}/S]R_g) \parallel ([\{N_0, N_1, N_2\}/S]R_g
\]

which abbreviates:

\[
r(\text{full}, z) :
\]

\[
\{\langle [\{N_0, N_1, N_2\}/S]R_g, [\{N_0, N_1\}/S]R_g, [\{N_1, N_2\}/S]R_g \rangle, \\
\langle [\{N_0, N_1\}/S]R_g, [\{N_0, N_1, N_2\}/S]R_g, [\{N_1, N_2\}/S]R_g \rangle, \\
\langle [\{N_0, N_1\}/S]R_g, [\{N_0, N_1\}/S]R_g, [\{N_0, N_1, N_2\}/S]R_g \rangle\}
\]
Example

\{x_0 = N_0 \land x_1 = N_1 \land x_2 = N_2\}

\begin{align*}
(\text{PUT}(x_0); \text{PUT}(x_1); \text{PUT}(x_2)) &\parallel \\
(\text{GET}(y_0); \text{GET}(y_1)) &\parallel \text{GET}(y_2)
\end{align*}

\{y_0 \in \{N_0, N_1\} \land y_1 \in \{N_1, N_2\} \land y_2 \in \{N_0, N_1, N_2\}\}

With

\begin{align*}
r(\text{full}, z) : (\lbrack N_0/X \rbrack R_p; \lbrack N_1/X \rbrack R_p; \lbrack N_2/X \rbrack R_p) &\parallel \\
(\lbrack\{N_0, N_1\}/S \rbrack R_g; \lbrack\{N_1, N_2\}/S \rbrack R_g) &\parallel \lbrack\{N_0, N_1, N_2\}/S \rbrack R_g
\end{align*}
To show that the program doesn’t abort we need:

\[
\{f_{ull} = 0\}
\]

\[
([N_0/X] R_p \ [N_1/X] R_p; [N_2/X] R_p) \parallel
\]

\[
([\{N_0, N_1\}/S] R_g; [\{N_1, N_2\}/S] R_g) \parallel
\]

\[
[\{N_0, N_1, N_2\}/S] R_g
\]

\[
\{f_{ull} = 0\}
\]
We define composition of relations:

\[(x, z) \in R_0 \circ R_1 \equiv (\forall y'. (x, y') \in R_0 \Rightarrow y' \in \text{dom}(R_1)) \land \exists y. (x, y) \in R_0 \land (y, z) \in \overline{R_1}\]

and we define \(\{P\} S \{Q\}\) by:

\[\forall \langle R_0, \ldots, R_n \rangle \in S. \ P \subseteq \text{dom}(R_0 \circ R_1 \circ \cdots \circ R_n) \land \{\sigma' \mid \exists \sigma \in P. (\sigma, [\sigma']) \in (R_0 \circ R_1 \circ \cdots \circ R_n)\} \subseteq Q\]
Conclusion

- In CSL, reusable specification are hard to find.
- Using semantic methods in a logic will ease this problem.
Related Work

Closely related:

- Steve Brookes: A Semantics for Concurrent Separation Logic.
- Peter O’Hearn: Resources, Concurrency and Local Reasoning.

More remotely related:

- Gotsman, Cook, Parkinson, and Vafeiadis: Proving that non-blocking algorithms don’t block.
- Feng: Local Rely-Guarantee Reasoning.