Chapter 10

Safety for substructural specifications

In Chapter 9, we showed how the preservation theorem could be established for a wide variety of SSOS semantics, both ordered abstract machines and destination-passing style semantics. The methodology of generative invariants we espoused goes significantly beyond previous work on type preservation for operational semantics specifications in substructural logic. Neither Linear LF encodings by Pfenning, Cervesato, and Reed [CP02, Ree09], nor the Ordered LF encodings of Felty and Momigliano [FM12], discussed preservation for concurrent specifications or for first-class continuations.

More fundamentally, however, this previous work does not even provide a *language* for talking about progress theorems, the other critical companion of type safety theorems. These previous approaches were universally based on complete derivations. These complete derivations have the flavor of derivations in a big-step semantics, and it is difficult or even impossible to talk about progress for such specifications. The purpose of this chapter is to establish that the SLS framework's traces T and steps S, which correspond to partial proofs, provide a suitable basis for stating progress theorems (and therefore language safety theorems) and for proving these theorems.

We do not discuss progress and safety for the full range of specifications from Part II or Chapter 9, however. Instead, we will just discuss progress for two examples: the ordered abstract machine specification with parallelism and failure used as an example in Figure 9.1, and the extension of this specification with mutable storage. The rest is left for future work, though we claim that these two examples serve to get across all the concepts necessary to prove progress theorems for SSOS specifications. Ultimately, it is not only possible to prove progress and safety theorems using SSOS specifications in SLS; it's also reasonably straightforward.

10.1 Backwards and forwards through traces

In the last chapter, we worked on traces exclusively by induction and case analysis on the *last* steps of a generative trace. This form of case analysis and induction on the last steps of a trace can also be used to prove progress for sequential SSOS specifications, and it is actually necessary to prove progress for SSOS specifications with first-class continuations (discussed in Section 7.2.4 and Section 9.6) in this way, though we leave the details of this argument as future work. However, for the ordered abstract machine example from Figure 9.1, the other direction is more natural: we work by induction and case analysis on the *first* steps of a generative trace. The branching structure introduced by parallel continuation frames (that is, ordered propositions $\mathsf{cont2}\ f$) is what makes it more natural to work from the beginning of a generative trace, rather than the end.

The proof of progress relies critically on one novel property: that transitions in the generative trace do not tamper with terminals. Formally, we need to know that if $\Theta\{\Delta\} \leadsto^*_{\Sigma Gen} \Delta'$ under some generative signature ΣGen and if Δ contains only terminals, then there is some Θ' such that the final state Δ' matches $\Theta'\{\Delta\}$. We will implicitly use this property in most of the cases of the progress theorem below.

This property holds for all the generative signatures in Chapter 9, but establishing this property for generative signatures *in general* necessitates a further restriction of what counts as a generative signature (Definition 9.1). To see why, let $\Delta = (x_1{:}\langle \mathsf{retn}\ v\rangle\ ord, x_2{:}\langle \mathsf{cont}\ f\rangle\ ord)$ and consider the generative rule $\forall e.\{\mathsf{eval}\ e\}$, which is allowed under Definition 9.1. This rule could "break" the context by dropping an ordered $\mathsf{eval}\ e$ proposition in between x_1 and x_2. A sufficient general condition for avoiding this problem is to demand that any generative rule that produces ordered atomic propositions mentions an ordered nonterminal as a premise. (This is related to the property called *separation* in [SP08].)

10.2 Progress for ordered abstract machines

The progress property is that, if $T :: (x_0{:}\langle \mathsf{gen}\ tp_0\rangle\ ord) \leadsto^*_{\Sigma_{Gen9.4}} \Delta$ and $\Delta ↯$, then one of these three possibilities hold:

1. $\Delta \leadsto \Delta'$ under the signature from Figure 9.1,
2. $\Delta = y{:}\langle \mathsf{retn}\ v\rangle\ ord$, where v is a value, or
3. $\Delta = y{:}\langle \mathsf{error}\rangle\ ord$.

This is exactly the form of a traditional progress theorem: if a process state is well typed, it either takes a step under the dynamic semantics or is a final state (terminating with an error or returning a value).

The presence of parallel evaluation in Figure 9.1 necessitates that we generalize our induction hypothesis. The statement above is a straightforward corollary of Theorem 10.1 below.

Theorem 10.1 (Progress for ordered abstract machines)**.** *If* $T :: \Theta\{x{:}\langle \mathsf{gen}\ tp\rangle\ ord\} \leadsto_{\Sigma_{Gen9.4}} \Delta$ *and* $\Delta ↯$*, then either*

* $\Delta \leadsto \Delta'$ *under the signature from Figure 9.1 for some* Δ'*, or else*
* $T = (T_1; \{y\} \leftarrow \mathsf{gen/retn}\ tp\ v\ x; T_2)$ *and* $\cdot \vdash N : \mathsf{value}\ v\ true$*, or else*
* $T = (T_1; \{y\} \leftarrow \mathsf{gen/error}\ tp\ x; T_2)$.

In the proof of Theorem 10.1, we will not detail the parts of the proof that already arise in traditional proofs of progress for abstract machines. These missing details can be factored into two lemmas. The first lemma is that if $\cdot \vdash N : \mathsf{of}\ e\ tp\ true$, then the process state $\Theta\{x{:}\langle \mathsf{eval}\ e\rangle\ ord\}$ can always take a step; this lemma justifies the classification of eval as an *active* proposition as described in Section 7.2.2 and in [PS09]. The second lemma is traditionally called a *canonical forms* lemma: it verifies, by case analysis on the structure of typing derivations and values, that well-typed values returned to a well-typed frames can always take a step.

Proof. By induction and case analysis on the first steps of T. We cannot have $T = \diamond$, because we cannot apply restriction to a context containing the nonterminal $\mathsf{gen}\ \mathit{tp}$. So $T = S; T'$, and either $x \notin {}^{\bullet}S$ or $x \in {}^{\bullet}S$.

If $x \notin {}^{\bullet}S$, then $T' :: \Theta'\{x{:}\langle \mathsf{gen}\ \mathit{tp}\rangle\ \mathit{ord}\} \leadsto_{\Sigma_{Gen9.4}} \Delta$ and we can succeed by immediate appeal to the induction hypothesis.

If $x \in {}^{\bullet}S$, then we perform case analysis on the possible transitions enabled by $\Sigma_{Gen9.4}$:

* $S = \{y\} \leftarrow \mathsf{gen/eval}\ e\ \mathit{tp}\ (x \bullet !N)$ where $\cdot \vdash N : \mathsf{of}\ e\ \mathit{tp}\ \mathit{true}$.
  Because eval is a terminal, $\Delta = \Theta'\{y{:}\langle \mathsf{eval}\ e\rangle\ \mathit{ord}\}$, and we proceed by case analysis on N to show that the derivation can always take a step (eval is an active proposition).

* $S = \{y\} \leftarrow \mathsf{gen/retn}\ \mathit{tp}\ v\ (x \bullet !N \bullet !N_v)$ – succeed immediately.

* $S = \{y\} \leftarrow \mathsf{gen/error}\ \mathit{tp}\ x$ – succeed immediately.

* $S = \{y_1', y_2\} \leftarrow \mathsf{gen/cont}\ \mathit{tp}\ f\ \mathit{tp}'\ (x \bullet !N)$ where $\cdot \vdash N : \mathsf{off}\ f\ \mathit{tp}'\ \mathit{tp}\ \mathit{true}$.
  Invoke the i.h. on $T' : \Theta\{y_1'{:}\langle \mathsf{gen}\ \mathit{tp}'\rangle\ \mathit{ord},\ y_2{:}\langle \mathsf{cont}\ f\rangle\ \mathit{ord}\} \leadsto_{\Sigma_{Gen9.4}} \Delta$, and then perform case analysis on the result to prove that $\Delta \leadsto \Delta'$:
  - If $\Delta \leadsto \Delta'$, then we're done.
  - If $T' = (T_1'; \{y_1\} \leftarrow \mathsf{gen/retn}\ \mathit{tp}'\ v\ (y_1' \bullet !N' \bullet !N_v'); T_2')$,
    then because retn and cont are terminals, $\Delta = \Theta'\{y_1{:}\langle \mathsf{retn}\ v\rangle\ \mathit{ord},\ y_2{:}\langle \mathsf{cont}\ f\rangle\ \mathit{ord}\}$, and we proceed by simultaneous case analysis on N, N', and N_v' (canonical forms lemma).
  - If $T' = (T_1'; \{y_1\} \leftarrow \mathsf{gen/error}\ \mathit{tp}'\ y_1'; T_2')$,
    then because error and cont are terminals, $\Delta = \Theta'\{y_1{:}\langle \mathsf{error}\rangle\ \mathit{ord},\ y_2{:}\langle \mathsf{cont}\ f\rangle\ \mathit{ord}\}$, and we have $\{z\} \leftarrow \mathsf{ev/error}\ f\ (y_1 \bullet y_2) :: \Delta \leadsto \Theta'\{z{:}\langle \mathsf{error}\rangle\ \mathit{ord}\}$.

* $S = \{y_1', y_2', y_3\} \leftarrow \mathsf{gen/cont2}\ \mathit{tp}\ f\ \mathit{tp}_1\ \mathit{tp}_2\ (x \bullet !N)$ where $\cdot \vdash N : \mathsf{off2}\ f\ \mathit{tp}_1\ \mathit{tp}_2\ \mathit{tp}\ \mathit{true}$.
  Invoke the i.h. twice on $T' : \Theta\{y_1'{:}\langle \mathsf{gen}\ \mathit{tp}_1\rangle\ \mathit{ord},\ y_2'{:}\langle \mathsf{gen}\ \mathit{tp}_2\rangle\ \mathit{ord},\ y_3{:}\langle \mathsf{cont2}\ f\rangle\ \mathit{ord}\}$, once to see what happens to y_1', and another time to see what happens to y_2', and then perform case analysis on the result to prove that $\Delta \leadsto \Delta'$:
  - If either invocation returns the first disjunctive possibility, that $\Delta \leadsto \Delta'$, then we're done.
  - If both invocations return the second disjunctive possibility, then T' contain two steps
    $\{y_1\} \leftarrow \mathsf{gen/retn}\ \mathit{tp}_1\ v_1\ (y_1' \bullet !N_1 \bullet !N_{v1})$ and
    $\{y_2\} \leftarrow \mathsf{gen/retn}\ \mathit{tp}_2\ v_2\ (y_2' \bullet !N_2 \bullet !N_{v2})$. Because retn and cont2 are terminals,
    $\Delta = \Theta'\{y_1{:}\langle \mathsf{retn}\ v_1\rangle\ \mathit{ord},\ y_2{:}\langle \mathsf{retn}\ v_2\rangle\ \mathit{ord},\ y_3{:}\langle \mathsf{cont2}\ f\rangle\ \mathit{ord}\}$, and we proceed by simultaneous case analysis on N, N_1, N_{v1}, N_2, and N_{v2} (canonical forms lemma).
  - In all the remaining cases, one of the subcomputations becomes an error and the other one becomes another error or a returned value. In any of these cases, $\Delta \leadsto \Delta'$ by one of the rules ev/errret, ev/reterr, or by ev/errerr.

∗ $S = \{y_1', y_2\} \leftarrow \mathsf{gen/handle}\, tp\, e_2\, (x \bullet !N)$.
Invoke the i.h. on $T' : \Theta\{y_1'{:}\langle \mathsf{gen}\, tp' \rangle\, ord,\ y_2{:}\langle \mathsf{handle}\, e_2 \rangle\, ord\} \leadsto_{\Sigma_{Gen9.4}} \Delta$, and then perform case analysis on the result to prove that $\Delta \leadsto \Delta'$:

- If $\Delta \leadsto \Delta'$, then we're done.
- If $T' = (T_1'; \{y_1\} \leftarrow \mathsf{gen/retn}\, tp'\, v\, (y_1' \bullet !N' \bullet !N_v'); T_2')$,
then because retn and cont are terminals, $\Delta = \Theta'\{y_1{:}\langle \mathsf{retn}\, v \rangle\, ord,\ y_2{:}\langle \mathsf{cont}\, f \rangle\, ord\}$, and we have $\{z\} \leftarrow \mathsf{ev/catcha}\, v\, e_2\, (y_1 \bullet y_2) :: \Delta \leadsto \Theta'\{z{:}\langle \mathsf{retn}\, v \rangle\, ord\}$.
- If $T' = (T_1'; \{y_1\} \leftarrow \mathsf{gen/error}\, tp'\, y_1'; T_2')$,
then because error and cont are terminals, $\Delta = \Theta'\{y_1{:}\langle \mathsf{error} \rangle\, ord,\ y_2{:}\langle \mathsf{handle}\, e \rangle\, ord\}$, and we have $\{z\} \leftarrow \mathsf{ev/catchb}\, e_2\, (y_1 \bullet y_2) :: \Delta \leadsto \Theta'\{z{:}\langle \mathsf{eval}\, e_2 \rangle\, ord\}$.

This covers all possible first steps in the trace T, and thus completes the proof. □

10.3 Progress with mutable storage

Developing progress proofs to for stateful specifications requires a property that is dual to unique index sets (Definition 9.5, Section 9.5.1). Unique index sets require that there will be only ever be *at most one* proposition of a certain form, and the dual property, *assured index sets*, require that there is always *at least one* proposition of a certain form.

Definition 10.2. *A set S is an* assured index set *at a type τ under a generative signature Σ and an initial state $(\Psi; \Delta)$ if, whenever $(\Psi; \Delta) \leadsto_\Sigma^* (\Psi'; \Delta')$, then $\Psi \vdash_\Sigma t : \tau$ implies that, for some $\mathsf{a}/i \in S$, $x{:}\langle \mathsf{a}\, t_1 \ldots t_n \rangle\, lvl \in \Delta'$ where $t_i = t$.*

The set $\{\mathsf{gencell}/1, \mathsf{cell}/1\}$ is both a unique index set and an assured index set under $\Sigma_{Gen9.6}$ and the initial state $(\cdot; x_0{:}\langle \mathsf{gen}\, tp \rangle\, ord)$. The latter property is critical to finishing the extension of Theorem 10.1 proof in certain cases where we invoke the canonical forms lemma. When we invoked the canonical forms lemma in the cont branch of that proof, we started with the knowledge that $\Delta = \Theta'\{y_1{:}\langle \mathsf{retn}\, v \rangle\, ord,\ y_2{:}\langle \mathsf{cont}\, f \rangle\, ord\}$. Two new outcomes are introduced when we introduce mutable state as discussed in Section 6.5.1 and Section 9.4. The first is the possibility that $v = \mathsf{loc}\, l$ while $f = \mathsf{get1}$, and the second is the possibility that $f = \mathsf{set2}\, l$. In each case, we cannot proceed with rule ev/get1 or rule ev/set2, respectively, unless we also know that there is a variable binding $z{:}\mathsf{cell}\, l\, v'$ in Δ. We know precisely this because $\{\mathsf{gencell}/1, \mathsf{cell}/1\}$ is an assured index set, because $\Psi \vdash l{:}\mathsf{mutable\_loc}$, and because gencell propositions, as nonterminals, cannot appear in the generated process state Δ. Therefore, in the former case we can produce a step $\{y', z'\} \leftarrow \mathsf{ev/get1}\, l\, v'\, (y_1 \bullet y_2 \bullet z)$, and in the later case we can produce a step $\{y', z'\} \leftarrow \mathsf{ev/set2}\, v\, l\, v'\, (y_1 \bullet y_2 \bullet z)$.

10.4 Safety

We conclude by presenting the safety theorem for the ordered abstract machine specification from Figure 9.1. This theorem relates the encoding of the usual deductive formulation of the typing judgment, of $e\ tp$, to a progress property stated in terms of substructural process states.

Theorem 10.3 (Safety for ordered abstract machines). *If $T :: (x{:}\langle \mathsf{eval}\, e\rangle\ ord) \leadsto^* \Delta$ under the signature from Figure 9.1 and $\cdot \vdash N : \mathsf{of}\ e\ tp$, then either*

* *$\Delta \leadsto \Delta'$ under the signature from Figure 9.1 for some Δ', or else*
* *$\Delta = (y{:}\langle \mathsf{retn}\, v\rangle\ ord)$ and $\cdot \vdash N : \mathsf{value}\, v$, or else*
* *$\Delta = (y{:}\langle \mathsf{error}\rangle\ ord)$.*

Proof. First, by induction and case analysis on the last steps of T, we show that for all Δ' such that $T' :: (x{:}\langle \mathsf{eval}\, e\rangle\ ord) \leadsto^* \Delta'$ under the signature from Figure 9.1, we can construct a generative trace $T_g :: (x_0{:}\langle \mathsf{gen}\ tp\rangle\ ord) \leadsto^*_{\Sigma_{Gen9.4}} \Delta'$:

Base case $T' = \diamond$.
Construct $T_g = \{x\} \leftarrow \mathsf{gen/eval}\ tp\ e\ (x_0 \bullet !N) :: (x_0{:}\langle \mathsf{gen}\ tp\rangle\ ord) \leadsto^*_{\Sigma_{Gen9.4}} (x{:}\langle \mathsf{eval}\, e\rangle\ ord)$.

Inductive case $T' = T''; S$, where $T'' :: (x_0{:}\langle \mathsf{gen}\ tp\rangle\ ord) \leadsto^*_{\Sigma_{Gen9.4}} \Delta''$ and $S :: \Delta'' \leadsto_{\Sigma_{Gen9.4}} \Delta'$.
By the induction hypothesis, we have $T_g' :: (x_0{:}\langle \mathsf{gen}\ tp\rangle\ ord) \leadsto^*_{\Sigma_{Gen9.4}} \Delta''$. By preservation (Theorem 9.3) on T_g' and S, we have $T_g' :: (x_0{:}\langle \mathsf{gen}\ tp\rangle\ ord) \leadsto^*_{\Sigma_{Gen9.4}} \Delta'$.

This means, in particular, that we can construct $T_g :: (x_0{:}\langle \mathsf{gen}\ tp\rangle\ ord) \leadsto^*_{\Sigma_{Gen9.4}} \Delta$.

By the progress theorem (Theorem 10.1) on T_g, there are three possibilities:

* If $\Delta \leadsto \Delta'$, then we're done.
* If $T_g :: (T_1; \{y\} \leftarrow \mathsf{gen/retn}\ tp\ v\ (x_0 \bullet !N' \bullet !N_v'); T_2)$, then by a trivial case analysis on T_1 and T_2 we can conclude that both are empty and, therefore, that $\Delta = (y{:}\langle \mathsf{retn}\, v\rangle\ ord)$.
* If $T_g :: (T_1; \{y\} \leftarrow \mathsf{gen/error}\ tp\ x_0; T_2)$, then by a trivial case analysis on T_1 and T_2 we can conclude that both are empty and, therefore, that $\Delta = (y{:}\langle \mathsf{error}\rangle\ ord)$.

This concludes the proof of safety. □

Bibliography

[CP02] Iliano Cervesato and Frank Pfenning. A linear logical framework. *Information and Computation*, 179(1):19–75, 2002. 2.1, 3.3.3, 4.1, 4.1.1, 4.1.3, 4.4, 5.2, 9.1.3, 10

[FM12] Amy Felty and Alberto Momigliano. Hybrid: A definitional two-level approach to reasoning with higher-order abstract syntax. *Journal of Automated Reasoning*, 48(1):43–105, 2012. 5.2, 9.1.3, 10

[PS09] Frank Pfenning and Robert J. Simmons. Substructural operational semantics as ordered logic programming. In *Proceedings of the 24th Annual Symposium on Logic in Computer Science (LICS'09)*, pages 101–110, Los Angeles, California, 2009. 1.2, 2.1, 2.5, 2.5.2, 2.5.3, 3.6.1, 4.7.3, 5, 5.1, 6.5, 6.5.3, 7.2, 7.2.2, 10.2

[Ree09] Jason Reed. *A Hybrid Logical Framework*. PhD thesis, Carnegie Mellon University, 2009. 4.1, 4.1.2, 4.7.3, 5.2, 10

[SP08] Robert J. Simmons and Frank Pfenning. Linear logical algorithms. In *Proceedings of the International Colloquium on Automata, Languages and Programming, Track B (ICALP'08)*, pages 336–347. Springer LNCS 5126, 2008. 3.6.1, 8.1, 10.1