Compositional Synthesis via Exact Interface Abstraction

September 1, 2025

Abstract

We formalize a two–phase approach to program synthesis for products of domains, where the primitive operations partition into three disjoint sets: Σ_X that act only on the x–coordinate, Σ_Y that act only on the y–coordinate, and a small cross–set $\Sigma_\times$ that couples x into y but leaves x unchanged ("triangular" couplings). Phase 1 solves the independent coordinates in an abstract space that ignores cross operations. Phase 2 refines by enumerating a small, exactly parameterized interface that specifies where a fixed number of cross operations are inserted relative to the X–program (and, under a commuting hypothesis, independent of Y). We make the definitions precise, state the required commutation assumptions explicitly, give a correct embedding from the abstraction to the concrete DSL, provide tight counting/complexity bounds, and replace an unsound parity check with a sound abstract–interpretation filter. A worked example illustrates the mechanics. The resulting math is self–contained and, under the stated assumptions, exact. Empirically, this two-phase method explores $\approx$ 4–59$\times$ fewer nodes (geometric mean across K) and achieves $\approx$ 8–184$\times$ speedups versus global search across benchmarks.

Keywords: compositional synthesis; exact abstraction; interface enumeration; commutation; abstract interpretation.

1 Setup and Assumptions

Let G be the state space with coordinates (x, y). We consider a DSL whose primitive operations are partitioned into pairwise disjoint sets

$$\Sigma = \Sigma_X \,\dot\cup\, \Sigma_Y \,\dot\cup\, \Sigma_\times.$$

Assumptions.

(A1) (*Coordinate restriction*) Each $u \in \Sigma_X$ updates only x, i.e., there exists f_u with $u(x, y) = (f_u(x), y)$. Each $v \in \Sigma_Y$ updates only y, i.e., there exists g_v with $v(x, y) = (x, g_v(y))$.

(A2) (*Triangular cross–ops*) Each $\kappa \in \Sigma_\times$ leaves x unchanged and may update y as a function of x: there exists h_κ and a binary operation $\oplus$ on the y–domain such that

$$\kappa(x, y) \;=\; (x,\; y \oplus h_\kappa(x)).$$

(A3) (*Commutation*) For all $u \in \Sigma_X$ and $v \in \Sigma_Y$, $u \circ v \equiv v \circ u$ (they act on disjoint coordinates). Moreover, for all $v \in \Sigma_Y$ and $\kappa \in \Sigma_\times$, $v \circ \kappa \equiv \kappa \circ v$. This holds, for example, when v is an additive update on y and $\oplus$ is addition.

A (concrete) program is a word $w \in \Sigma^*$; its denotation $[\![w]\!] : G \to G$ is given by functional composition left to right. We write $\pi_X(x, y) = x$ and $\pi_Y(x, y) = y$ for the coordinate projections.

A training dataset is a finite set $D = \{(s_i, t_i)\}_{i=1}^N \subseteq G \times G$. The synthesis task is to find $w \in \Sigma^*$ such that $[\![w]\!](s_i) = t_i$ for all i.

2 Phase 1: Base Abstraction A

Define the abstract space

$$A \;=\; \Sigma_X^* \times \Sigma_Y^*.$$

An element $a = (p_X, p_Y) \in A$ specifies an X–only program $p_X \in \Sigma_X^*$ and a Y–only program $p_Y \in \Sigma_Y^*$.

Embedding. Because of (A1)–(A3) on Σ_X and Σ_Y, any interleaving of p_X and p_Y is semantically equivalent. Thus the embedding

$$e : A \to \Sigma^*, \qquad e(p_X, p_Y) = \text{any interleaving of } p_X \text{ and } p_Y$$

is well–defined up to denotational equivalence: $[\![e(p_X, p_Y)]\!]$ does not depend on which interleaving is chosen.

Solving in A. Project the dataset and synthesize independently:

$$\begin{aligned}&\text{find } p_X \in \Sigma_X^* \text{ s.t. } \forall i,\ \pi_X([\![p_X]\!](s_i)) = \pi_X(t_i),\\ &\text{find } p_Y \in \Sigma_Y^* \text{ s.t. } \forall i,\ \pi_Y([\![p_Y]\!](s_i)) = \pi_Y(t_i).\end{aligned}$$

If both succeed, $(p_X, p_Y) \in A$ is a base solution. In general, cross operations are needed; we add them compositionally next.

3 Phase 2: Interface Abstraction A^+

Fix $K \in \mathbb{N}$ and a designated $\kappa \in \Sigma_\times$ (extensions to multiple cross–ops are straightforward by labeling). For $L_X \in \mathbb{N}$, define the *slot interface*

$$\Pi_{L_X,K} \;=\; \Big\{ \alpha \in \{0, 1, \ldots, L_X\}^K \;\Big|\; \alpha_1 \le \alpha_2 \le \cdots \le \alpha_K \Big\}.$$

For $\alpha \in \Pi_{L_X,K}$, let $m_i(\alpha) = |\{j : \alpha_j = i\}|$ for $i = 0, \ldots, L_X$. Given $p_X = (x_1, \ldots, x_{L_X}) \in \Sigma_X^{L_X}$, define the interleaving word

$$\text{interleave}_\alpha(p_X, \kappa^K) \;=\; \kappa^{m_0}\, x_1\, \kappa^{m_1}\, x_2 \cdots x_{L_X}\, \kappa^{m_{L_X}}. \tag{1}$$

We now enrich A by recording where the K cross–ops occur relative to p_X while keeping p_Y explicit:

$$A^+ \;=\; \bigcup_{L_X, L_Y \ge 0} \Sigma_X^{L_X} \times \Sigma_Y^{L_Y} \times \Pi_{L_X,K}. \tag{2}$$

The concrete embedding refines e:

$$e^+(p_X, p_Y, \alpha) \;=\; \text{interleave}_\alpha(p_X, \kappa^K) \;\cdot\; p_Y. \tag{3}$$

By (A3), p_Y commutes with κ, so placing all Y–ops at the end does not lose generality.

Statement (Completeness for triangular cross–ops). Under Assumptions (A1)–(A3), if a concrete target is an interleaving of some $p_X \in \Sigma_X^{L_X}$, some $p_Y \in \Sigma_Y^{L_Y}$, and K copies of $\kappa \in \Sigma_\times$, then there exists $\alpha \in \Pi_{L_X,K}$ such that $e^+(p_X, p_Y, \alpha)$ is denotationally equivalent to the target on all inputs.

Family size and counting. For fixed lengths (L_X, L_Y) and fixed κ:

$$|\Pi_{L_X,K}| = \binom{L_X + K}{K} \qquad \text{(stars and bars)}, \quad (4)$$

$$|\{\text{all interleavings of lengths } L_X, L_Y, K\}| = \binom{L_X + L_Y + K}{L_X,\ L_Y,\ K} m_X^{L_X}\, m_Y^{L_Y}, \quad (5)$$

where $m_X = |\Sigma_X|$ and $m_Y = |\Sigma_Y|$. Thus, for fixed (p_X, p_Y), the interface search enumerates exactly $\binom{L_X+K}{K}$ candidates instead of the larger multinomial family in (5). If $|\Sigma_\times| > 1$ and different cross–ops may occupy slots, multiply by $m_\times^K$ and add a label sequence to the interface.

4 Complexity Comparison

Let b be a generic branching factor for the global DSL and b_X for the X–only DSL. To reach a minimal solution of lengths (L_X, L_Y, K):

- **Global baseline (flat search):** time $\Theta(b^{L_X+L_Y+K})$.
- **Compositional (this paper):** first solve X in $\Theta(b_X^{L_X})$ (and Y similarly if needed), then enumerate $\binom{L_X+K}{K}$ interfaces and evaluate them on D. The refinement cost is $\Theta\big(\binom{L_X+K}{K} \cdot C_D\big)$, where C_D is the cost of evaluating one candidate on the dataset.[1]

5 Worked Example

Let the domains be integers with addition. Define primitives

$$\begin{aligned}
\Sigma_X &:= \{\texttt{inc_x} : (x,y) \mapsto (x{+}1, y),\ \texttt{double_x} : (x,y) \mapsto (2x, y)\},\\
\Sigma_Y &:= \{\texttt{inc_y} : (x,y) \mapsto (x, y{+}1)\},\\
\kappa &\in \Sigma_\times, \qquad \kappa(x,y) = (x, y{+}x).
\end{aligned}$$

Assumptions (A1)–(A3) hold: κ leaves x unchanged; Y–ops commute with κ since both are additive on y.

Take

$$\begin{aligned}
p_X^\star &= [\texttt{inc_x},\ \texttt{inc_x},\ \texttt{double_x},\ \texttt{inc_x}] \quad (L_X{=}4),\\
p_Y^\star &= [\texttt{inc_y},\ \texttt{inc_y}] \quad (L_Y{=}2),\\
K &= 1, \quad \alpha^\star = (2).
\end{aligned}$$

Then, by (3),

$$w^\star \ = \ e^+(p_X^\star, p_Y^\star, \alpha^\star) \ = \ \texttt{inc\_x},\ \texttt{inc\_x},\ \kappa,\ \texttt{double\_x},\ \texttt{inc\_x},\ \texttt{inc\_y},\ \texttt{inc\_y}.$$

Its denotation is

$$\begin{aligned}
x' &= 2(x{+}2) + 1 \ = \ 2x + 5,\\
y' &= y + (x{+}2) + 2 \ = \ y + x + 4,
\end{aligned}$$

which matches the intended behavior.

The full family of interleavings for $(L_X, L_Y, K) = (4, 2, 1)$ has size $\binom{7}{4,2,1} = 105$; for fixed $(p_X^\star, p_Y^\star)$, the interface search considers only $\binom{4+1}{1} = 5$ placements of κ.

[1] If multiple p_X (or p_Y) candidates survive Phase 1, multiply by that number.

6 Parity–Aware Extension A^{++}

Sometimes we require a semantic side condition, e.g. "the output y is even whenever the input y is even." The following check is *sound and exact* for parity properties and replaces any single–input heuristic tests.

Parity abstraction. Let the abstract state space be $\{0,1\}^2$, recording the parity of (x,y). Each primitive $s \in \Sigma$ induces a function $\widehat{s} : \{0,1\}^2 \to \{0,1\}^2$ computed from its definition modulo 2 (e.g. $\widehat{\texttt{inc\_x}}(p_x,p_y) = (p_x \oplus 1, p_y)$, $\widehat{\kappa}(p_x,p_y) = (p_x, p_y \oplus p_x)$). Extend to words by composition: $\widehat{w} = \widehat{s_1} \circ \cdots \circ \widehat{s_\ell}$.

Property check. Encode the desired parity property as a set $\Phi \subseteq \{0,1\}^2$ of allowed output parities for each allowed input parity (e.g. "preserve even y" means: if $(\_,0)$ is input then $(\_,0)$ must be output). The candidate w passes the filter iff for every $(p_x,p_y) \in \{0,1\}^2$ that satisfies the precondition, $\widehat{w}(p_x,p_y)$ satisfies the postcondition. Since the domain has only four abstract states, the check is constant time per candidate.

Remarks. This analysis is independent of the interface trick and can be applied as a final filter to any $e^+(p_X,p_Y,\alpha)$. If the set of allowed primitives must themselves be parity–preserving, simply restrict Σ to the subset whose $\widehat{s}$ preserves the property; closure under composition then follows immediately.

7 Experimental Results

Setup. We evaluate on a suite of synthesis tasks over product domains with triangular cross operations (Sec. 3). Baselines are flat global search (uniform cost) and a goal-directed enumerative search. We measure (i) explored nodes and (ii) wall-clock time. In this suite we set $L_Y{=}0$, so the minimal target length equals $L_X{+}K$.

Metrics. "Nodes" counts program states visited by the searcher; "Time" is measured in seconds on the same machine.

Headline results. Across our benchmarks, the two-phase approach (A then A^+) achieves *geometric-mean* node reductions of $\approx$ 4.21×, 19.29×, 29.21×, 58.70× for $K = 0,1,2,3$ respectively, and time speedups of $\approx$ 7.63×, 44.22×, 79.66×, 184.44×. On the hardest subset ($L_X{=}8$, $b{=}4$), node reductions range from $\approx$ 4.33× ($K{=}0$) to $\approx$ 82.83× ($K{=}3$).

Ablations. Removing interface refinement (A only) fails on tasks that require coupling; removing base factorization (A^+ only) regresses to global search.

Scaling with K. Fixing $L_X{+}L_Y$ and increasing the number of cross operations K, the refinement cost follows the predicted $\binom{L_X+K}{K}$ curve, whereas the global multinomial family grows as $\binom{L_X+L_Y+K}{L_X,L_Y,K}$ (Sec. 3). Empirically this gap explains most of the node and time reductions.

Per-benchmark summary.

K	Nodes× (gmean)	Time× (gmean)	Runs n
0	4.208019	7.629846	9
1	19.289877	44.219750	5
2	29.210168	79.663099	9
3	58.699133	184.438869	7

Table 1: Geometric-mean speedups computed directly from the implementation output of `scaling.py`.

K	Global nodes	Comp nodes	Nodes× (Global/Comp)	Setting
0	1609	372	4.327956	L_X=8, b=4
1	6384	374	17.071657	L_X=8, b=4
2	22569	380	59.392105	L_X=8, b=4
3	32061	387	82.829971	L_X=8, b=4

Table 2: Hard subset reported by the implementation (`scaling.py`): L_X=8, b=4, L_Y=0. Nodes ratios are computed from the printed values.

Implementation notes. Our current implementation mirrors the two-phase algorithm with a few pragmatic choices:

- **Early stop in Phase 2.** During interface refinement we enumerate slot patterns in $\Pi_{L_X,K}$ but *stop at the first* placement that satisfies the training spec. Thus the number of tested interfaces can be $< \binom{L_X+K}{K}$ in practice.
- **One-shot p_X.** Phase 1 returns the first p_X consistent with the X-projection of the data; if no interface placement works, we *do not backtrack* to alternate p_X's. This explains the 30/36 overall success rate reported by the script.
- **Global baseline dedup.** The global BFS baseline uses per-depth semantic deduplication (states with identical denotations at a given depth are not re-expanded). It remains exhaustive up to the target length.
- **No parity filter.** The parity-aware filter of Sec. 6 is *not* used in these experiments.

Both methods succeeded on **30/36** runs; aggregates above are computed over runs where both methods succeeded.

Table 1 reports geometric-mean speedups; details and scripts are provided in the repository.

8 Discussion and Limitations

Practical variant used in experiments. The experiments adopt a one-shot variant: we keep the first p_X returned by Phase 1 and stop Phase 2 as soon as a satisfying interface is found. Trying multiple p_X's would further increase success rates at the cost anticipated in Sec. 4 (multiplying the refinement cost by the number of p_X candidates).

The interface abstraction is exact under (A1)–(A3); if cross operations can *also* modify x, the interface must be extended (e.g. by tracking slots relative to both p_X and p_Y or by richer summaries). If Y–ops do not commute with κ, one can either bake their labels into the interface (track where Y occurs between slots) or keep them explicit in A^+ without commuting them to the end; the counting and complexity formulas then change accordingly.

Takeaway. Separating "what the coordinates do" from "where a small number of couplings go" yields a provably smaller, exactly parameterized search in Phase 2 while keeping full expressiveness for triangular couplings.