www.ens-lyon.eu/ www.cs.colostate.edu/ www.inria.fr/ www.cnrs.fr/ www.univ-lyon1.fr/ www.ens-lyon.fr/LIP/

Semantic Program Optimization

Avoiding (some) Data Dependencies

Introduction

- Tiling is an effective transformation for parallelism, locality improvement and granularity.
- However, its legality relies on not violating program dependencies.

☛ We propose to use semantic properties (associativity, commutativity) to do tiling.
☛ The semantically-equivalent tiled program has different dependencies.
☛ The derivation of this transformation relies on program equivalence checking.

Using Semantic Properties for Tiling

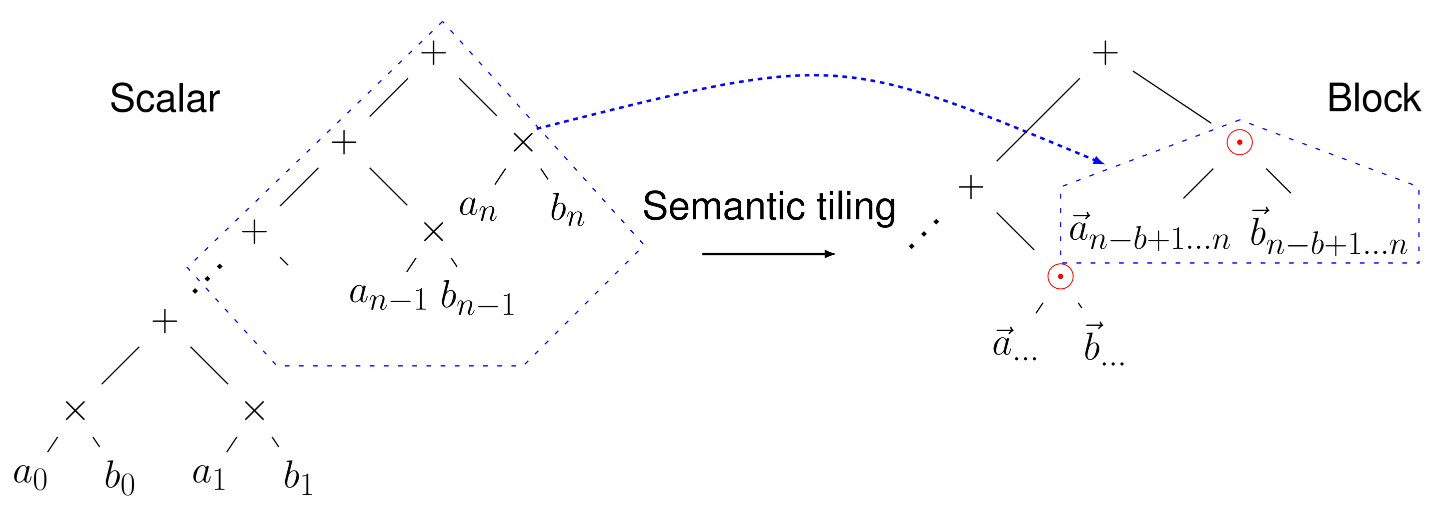
Semantic tiling: (for this poster) Transformation of a program operating on scalars into an equivalent program operating on blocks with the same structure.

☛ **"Semantic":** Operators are not the same (semantic rules are used)
☛ **"Tiling":** Operands on blocks instead of scalars (raise the level of abstraction)
$\Rightarrow$ Locality benefits (like in classical tiling).

Properties of semantic tiling:

- Reorganize operations and dependencies of a program
- Recursive call on a smaller instance $\rightarrow$ Divide-and-conquer scheme.
- Apply on linear algebra and graph theoretic algorithms (need a notion of block)
- Applicability conditions are different than for classical tiling. (semantic tilable but non-classically tilable: Algebraic Path Problem)

Semantically tilable examples: Matrix multiply, Forward substitution, LU, Cholesky, sub-problems of APP (shortest path in a graph), ...

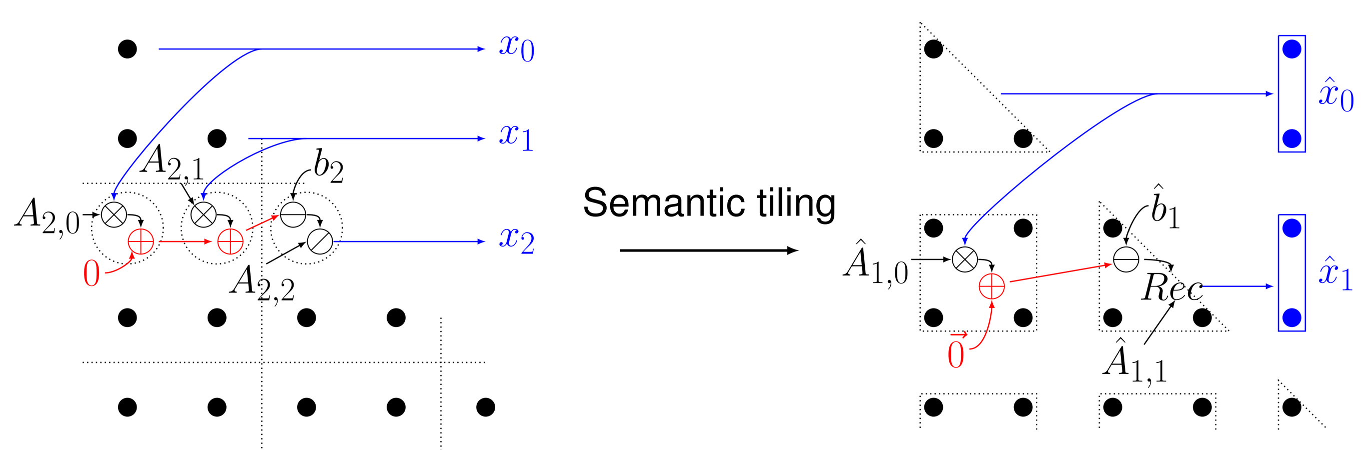


- Semantic tiling applied to scalar product -

Toward Automatic Derivation

Two approaches are being explored:

1. Using sophisticated version of "pattern matching" to hypothesize the structure of the blocked program. Then, proving its validity through program equivalence checking.
2. Transforming the scalar program into its semantic tiled version, by using rewriting rules based on algebraic properties. More precisely:
   a. **Data tiling** of the inputs, outputs and temporaries of the program.
   b. **Extracting the program slice** that touches the same data block.
   c. **Check the equivalence** with a corresponding matrix-level operator.
   d. **Substituting** them by the matrix-level operators $\rightarrow$ Semantic tiled program.



- Semantic tiling applied to forward substitution -

Checking Program Equivalence

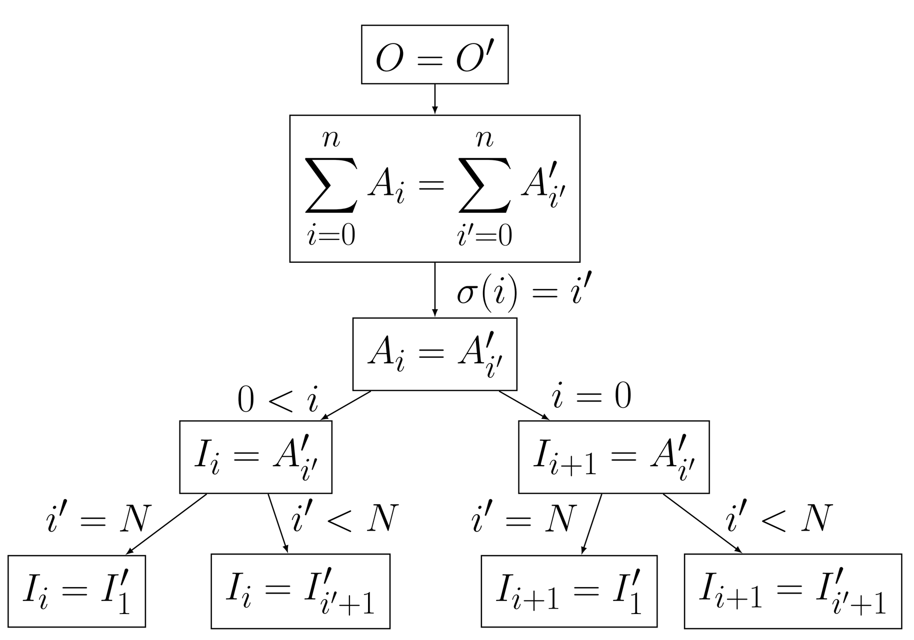
We build on the 2-step equivalence algorithm for SAREs proposed on [1]:

- Compile the program unification problem as an integer interpreted automaton.
- Then, check reachability by symbolic execution of this automaton.

Do not manage any semantic rules (such as associativity and commutativity).

$\Rightarrow$ We proposed [2] an extension of the equivalence algorithm to manage **reductions**, i.e. successive applications of an associative and commutative operator over a family of expressions (example: $\sum_{i=0}^{n-1} A[i]$).

Main idea: find a bijection σ between the reduction bodies.



$$\begin{cases} O = \sum_{i=0}^{n} A_i \\ A_i = \begin{cases} I_i & \text{when } 0 < i \\ I_{i+1} & \text{when } i = 0 \end{cases} \end{cases}$$

$$\begin{cases} O' = \sum_{i'=0}^{n} A'_{i'} \\ A'_{i'} = \begin{cases} I'_1 & \text{when } i' = N \\ I'_{i'+1} & \text{when } i' < N \end{cases} \end{cases}$$

$$\leadsto \sigma : \begin{cases} 0 \mapsto 0 \\ 1 \mapsto N \\ i \mapsto i-1 & \text{when } i \geq 2 \end{cases}$$

- Equivalence automaton and derived σ -

Conclusion

Semantic tiling is a semantic program transformation that raises the level of abstraction of a program.

$\Rightarrow$ Allows us to "break" some dependencies during tiling, while retaining program semantics.

$\Rightarrow$ Raises the granularity of the considered operations $\rightarrow$ Suitable for HLS.

Future Work

Semantic tiling automatic derivation:

- Mathematical formalization in progress. Encountered issues:
  - How to systematically group instructions together?
  - How to prune the higher-order operator?
  - How do we recognize an instruction block corresponding to a recurrence call?
- How to characterize the programs that can be semantically tiled ?
- Generalize beyond just blocking.

Program equivalence with reductions:

- Implementation in progress.
- Improve the applicability of the equivalence algorithm.
- Can be reused for other purpose (transformation/validation)

References

[1] *On the Equivalence of Two System of Affine Recurrence Equations* (by D.Barthou, P.Feautrier and X.Redon)

[2] *On Program Equivalence with Reductions* (by G.Iooss, C.Alias, S.Rajopadhye, technical report in preparation)

People Involved

Guillaume Iooss (ENS Lyon - CSU - PhD student)
guillaume.iooss[-at-]ens-lyon.fr

Christophe Alias (ENS Lyon/Inria - Faculty Member)
christophe.alias[-at-]ens-lyon.fr

Sanjay Rajopadhye (CSU - Faculty Member)
sanjay.rajopadhye[-at-]colostate.edu