# A Generic Framework for the Analysis and Specialization of Logic Programs

**Germán Puebla**[*], **Elvira Albert**[**], **and Manuel Hermenegildo**[*,***]

(∗)*Technical University of Madrid (Spain)*
(∗∗)*Complutense University of Madrid (Spain)*
(∗ ∗ ∗)*University of New Mexico (USA)*

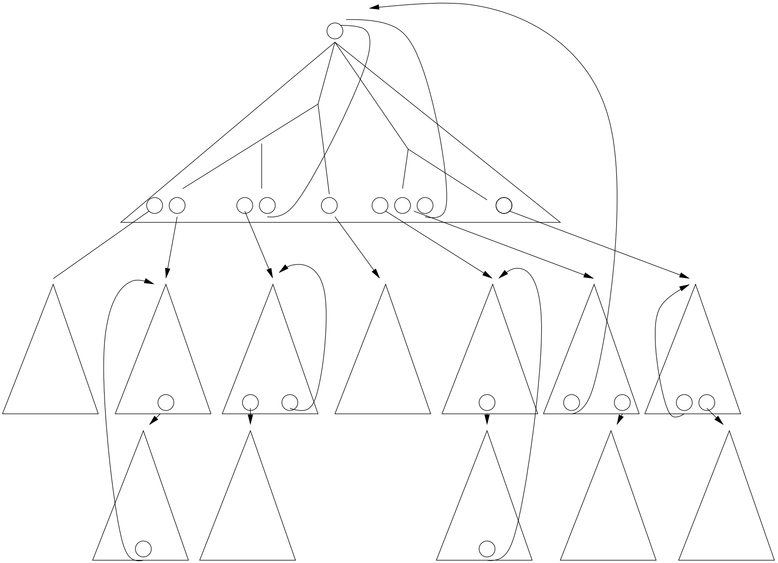
15th Workshop on Logic-based Methods In Programming Environments (WLPE'05)

Sitges, October 5, 2005

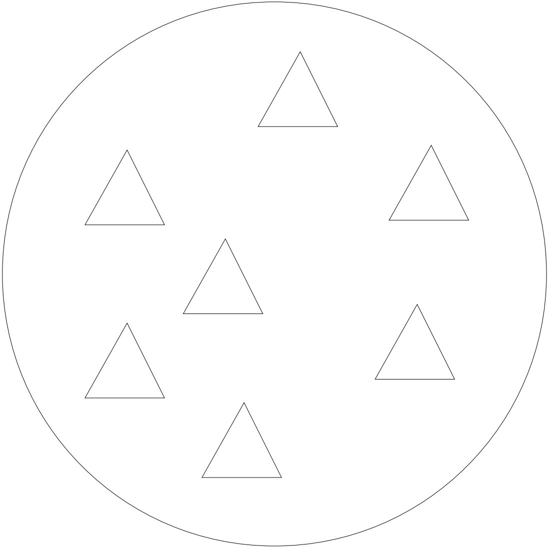
# Motivation

- Traditional partial evaluation of logic programs
  - Based on SLD semantics
  - Nice and simple
  - Agressive transformations
  - But sometimes is not very accurate!
- Traditional partial evaluation of logic programs
  - Based on And–Or trees
  - Well understood
  - Often accurate results
  - But sometimes is not very accurate!

# Partial Deduction and SLD-Trees



# Loss of Information in SLD-Trees



# Challenges in combining abstract information and unfolding

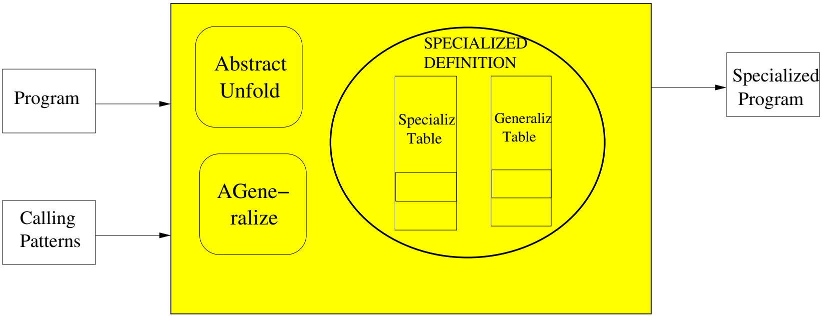
1. Exploiting abstract information to abstractly execute atoms which allows more unfolding
   - All calls to the tests $\texttt{ground}_{2,1}$`(X)` and $\texttt{var}_{2,2}$`(W)` will succeed
   - Calls to $\texttt{ground}_{8,1}$`(X)` will succeed, while calls to $\texttt{var}_{7,1}$`(X)` will fail
   - Groundness and freeness not sufficient to determine that, in $2^{nd}$ execution of `formula`, tests $\texttt{ground}_{2,1}$`(X)` and $\texttt{var}_{2,2}$`(W)` succeed.
2. Unfolding steps to prune away useless branches, which results in improved success information
   - On success of $\texttt{minus}_{2,4}$`(T,X,X2)`, `X2` not guaranteed to be ground ($\texttt{minus}_{6}$`/3` succeeds with `X2` variable)
   - However, for calls described by the `entry`, third clause for `minus/3` is useless, i.e., will never contribute to a success
   - Unfolding makes calls to `minus/3` sufficiently instantiated (third clause disregarded) and, thus, all its calls succeed with `X2` ground.

# Challenges in combining abstract information and unfolding

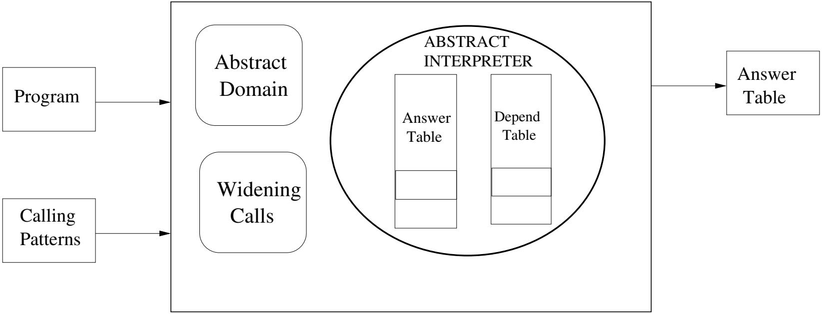
3. Propagating success information (fixpoint computations) simultaneously results in improved unfolding:
   - Need fixpoint computation to determine that, upon success of `twice`$_{2,5}$`(X2,W)` (thus success of `formula`$_{1,1}$`(X,W)`), `W` is ground.
   - Success substitution for `formula`$_{1,1}$`(X,X1)` is call substitution for `formula`$_{1,2}$`(X1,X2)`.
   - Success of test `ground`$_{2,1}$`(X)` (reachable from `formula`$_{1,2}$`(X1,X2)`) cannot be established unless we propagate success.
4. Having information on non *downwards-closed* properties
   - Whenever we call `formula(X,W)`, `W` is a variable
   - This property cannot be captured if we restrict ourselves to downwards-closed domains.

- Our framework is able to abstractly execute all calls to mode tests `ground/1` and `var/1`, and predicates `two/1` and `minus/3` are both fully unfolded and no longer appear in the residual code.

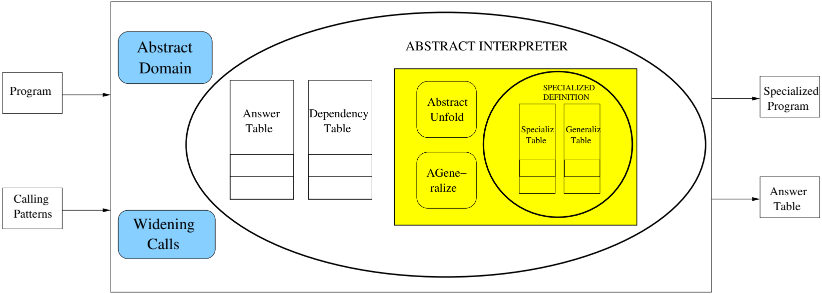
# Partial Evaluation



# Abstract Interpretation



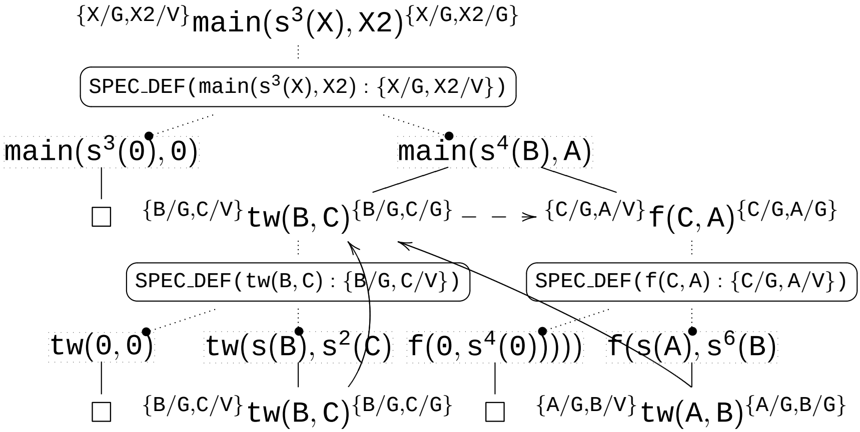
# Abstract Interpretation with Specialized Definitions



# Integration of abstract interpretation and partial deduction

- Previous (partial) integrations starting from both the partial deduction and abstract interpretation perspectives.
- Proposal: first fully described generic algorithm for efficient and precise integration from an abstract interpretation perspective.
- Starting point: state-of-the-art algorithms for context-sensitive, polyvariant abstract interpretation and (abstract) partial deduction
- Key ingredients: combining the best of both worlds:
  1. accurate success propagation inherent to abstract interpretation
  2. powerful program transformations achievable by partial deduction
- Specialized definitions: calls in analysis graph are not analyzed w.r.t. original definition of procedures but w.r.t. specialized definitions
  - specialized definitions obtained by unfolding and abstract executability.
- Benefits:
  1. Different combinations of parameters correspond to existing algorithms for program analysis and specialization.
  2. Strictly more precise results than individual techniques.
- Proposed algorithm: a key component of the `CiaoPP` system.

# Analysis Graph for the Example



# Generic framework for analysis and specialization

- Generic framework for analysis and specialization of LP: currently the basis of the analysis/specialization system implemented in the `CiaoPP` preprocessor
- Versatility can be seen by recasting well-known specialization and analysis frameworks as instances:
  - **Polyvariant AI**: Our algorithm can behave as Polyvariant AI by defining:
    - ⋆ *AGeneralize* operator which returns always the base form of an expression
    - ⋆ *AUnfold* operator which performs a single derivation step
  - **Multivariant AS:** The specialization power of abstract specialization can be obtained by using:
    - ⋆ the same *AGeneralize* described above
    - ⋆ *AUnfold* operator which always performs a derive step followed by zero or more abstract execution steps.

# Generic framework for analysis and specialization

- **Classical PD:** Our method can be used to perform classical PD by using:
  - an abstract domain with the single abstract value $\top$
  - the identity function as *Widen_Call* rule
- **APD:** Several approaches have been proposed which extend PD by using abstract substitutions.
  - They either fail to do so or propose means for propagating success information which are not fully integrated with the APD algorithm
  - These proposals are either strongly coupled to a particular (downward closed) abstract domain or do not provide the exact description of operations on the abstract domain which are needed by the framework, other than general correctness criteria.

# Conclusions

- Novel scheme for a seamless integration of the techniques of abstract interpretation and partial deduction.
- Parametric w.r.t. the abstract domain and the control issues which guide the partial deduction process.
- Existing proposals use AI as a *means* for improving PD rather than as a *goal*. Thus, their objective is to yield a PD rather than to compute a safe approximation of its success.
- Unlike them, our main objective is to improve success information by analyzing the specialized code, rather than the original one.
- Achieved by smoothly *interleaving* both techniques which improves success information.
- With more accurate success information, we can improve further the quality of partial evaluation.
- The overall method thus yields not only a specialized program but also a safe approximation of its behaviour.