30 Years of AI (1977-2007):

Development of Safe and Efficient Programs using Abstract Interpretation

Manuel Hermenegildo

E. Albert, P. Arenas, F. Bueno Carrillo, D. Cabeza, M. Carro, A. Casas, M. Codish, J. Correas, S.K. Debray, M. García de la Banda, S. Genaim, N.-W. Lin, P. López-García, K. Marriott, E. Mera, J. Morales, K. Muthukumar, M. Méndez-Lojo, J. Navas, P. Pietrzak, G. Puebla, F. Rossi, P. Stuckey, C. Vaucheret, R. Warren, D. Zanardini, M. Alpuente, M. Comini, Escobar, M. Falaschi, S. Lucas, M. Gallardo, E. Pimentel

CS Dept., T.U. Madrid, Spain
CS Dept., U. of Texas at Austin, USA
CS Dept., Complutense U. Madrid, Spain
CS Dept., U. of Valencia, Spain
CS and EECE Depts., U. of New Mexico, USA
Microel. and Computer Tech. Corp. (MCC), USA
CS Dept., U. of Malaga, Spain
IMDEA SW Development Technology Institute, Spain

~~30~~ 22 Years of AI (~~1977~~ 1985-2007):

Development of Safe and Efficient Programs using Abstract Interpretation

Manuel Hermenegildo

E. Albert, P. Arenas, F. Bueno Carrillo, D. Cabeza, M. Carro, A. Casas, M. Codish, J. Correas, S.K. Debray, M. García de la Banda, S. Genaim, N.-W. Lin, P. López-García, K. Marriott, E. Mera, J. Morales, K. Muthukumar, M. Méndez-Lojo, J. Navas, P. Pietrzak, G. Puebla, F. Rossi, P. Stuckey, C. Vaucheret, R. Warren, D. Zanardini, M. Alpuente, M. Comini, Escobar, M. Falaschi, S. Lucas, M. Gallardo, E. Pimentel

CS Dept., T.U. Madrid, Spain
CS Dept., U. of Texas at Austin, USA
CS Dept., Complutense U. Madrid, Spain
CS Dept., U. of Valencia, Spain
CS and EECE Depts., U. of New Mexico, USA
Microel. and Computer Tech. Corp. (MCC), USA
CS Dept., U. of Malaga, Spain
IMDEA SW Development Technology Institute, Spain

The Original Challenge / Early Steps: Parallelization

- The original problem: program parallelization (circa 1985 in US: UT Austin, MCC):
  - Detecting dependencies (pointer "sharing") among proc. calls, statements, etc.
  - Traditional approach (ad-hoc dataflow analysis [Chang, Despain, Degroot '85]) often incorrect — we wanted something *rigorous and more powerful*.
- Initial inspiration – [CC'77]? Nah... can't claim we understood CC'77 then. :-)
  - Actually, [Mycroft'80] "Abstract Interpretation and Optimizing Transformations for Applicative Programs" (but very FP oriented!)
  - [Mellish ICLP'86]: pointed the way (but not dealing w/ variable/pointer aliasing).
- Our first "obsession:" correctness/practicality –does AI really work?
  Built system [Warren, Debray, Herme. "MA3 system" ICSLP'88]:
  - Correct management of (variable/pointer) "aliasing" (correctness).
  - Top-down ("context sensitive").
  - Use of memo-tables (efficiency).
  - "Abstract compilation."
  - Actual implementation w/perf. results, connected to parallelizer –a first?

The Fixpoint Algorithm and the PLAI Generic Framework

- Bruynooghe's framework [87-91]: multivariance + genericity of framework –parametric on the domains– (but no tabling or efficient fixpoint).
- The fixpoint algorithm and PLAI analyzer: efficient, context sensitive, multivariant, parametric analysis! [Muthukumar, Herme. NACLP'89]
  - Tabling, multiple call–success pairs (efficient multivariance).
  - Dependency tracking to avoid recomputation —faster convergence.
  - Interprocedural, dealing with mutual recursion, etc. (project/extend).
- Many useful extensions (1990 on, Spain, mostly at UPM; also UNM):
  - Incremental framework [Herme, Puebla, Marriott, Stuckey ICLP'95, SAS'96, TOPLAS'00].
  - Extension to CLP [G. de la Banda, Herme., Bruynooghe, et al. ILPS'93, TOPLAS'96]
  - Extension to analysis of concurrent programs (dynamic scheduling) [G. de la Banda, Marriott, Herme., Stuckey POPL'94, ILPS'95, SAS'96],
  - Extension to Java/Java bytecode [Méndez-Lojo, Navas, Herme. BYTECODE'07, FTfJP 2007]. Intermediate representations [LOPSTR'07].

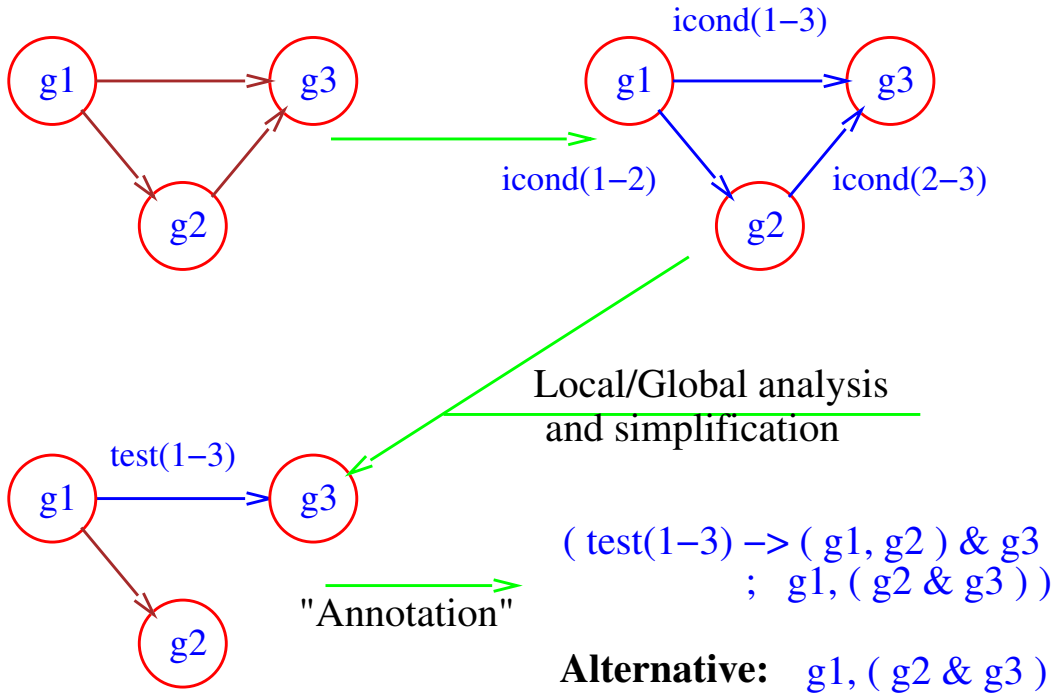
Some Achievements in Parallelization

- Fully automatic parallelizers for logic programs
  – arguably the most powerful parallelizers for "irregular" applications.
  [Herme. and Warren, CAN'87, Bueno, G. de la Banda, and Herme. ICLP'94, TOPLAS'99]
  - Parallelization using non-strict independence [Cabeza and Herme. SAS'94]
  - Parallelization of constraint programs [G. de la Banda and Herme. PLILP'96]
- Perhaps the first fully implemented, practical application of AI?
- Prompted considerable domain development:
  - Set sharing. [Jacobs-Langen NACLP 89, Muthukumar and Herme. NACLP'89 (abstr. unif.)]
  - Set sharing and freeness. [Muthukumar and Herme. ICLP'91]
  - Def (propositional horn clauses) [G. de la Banda and Herme. WSA'92]
  - Combinations with depth-k, shape analysis (regular types), etc.
  - Set sharing for Java [Méndez-Lojo, Herme. VMCAI'08]
- Prolog (Ciao, (C)LP) very useful languages: allow dealing with the most complex problems (pointers, irregular computations, irregular data, dynamic heap, dynamic dispatch, complex control flow, etc.) but in a much better structured context.

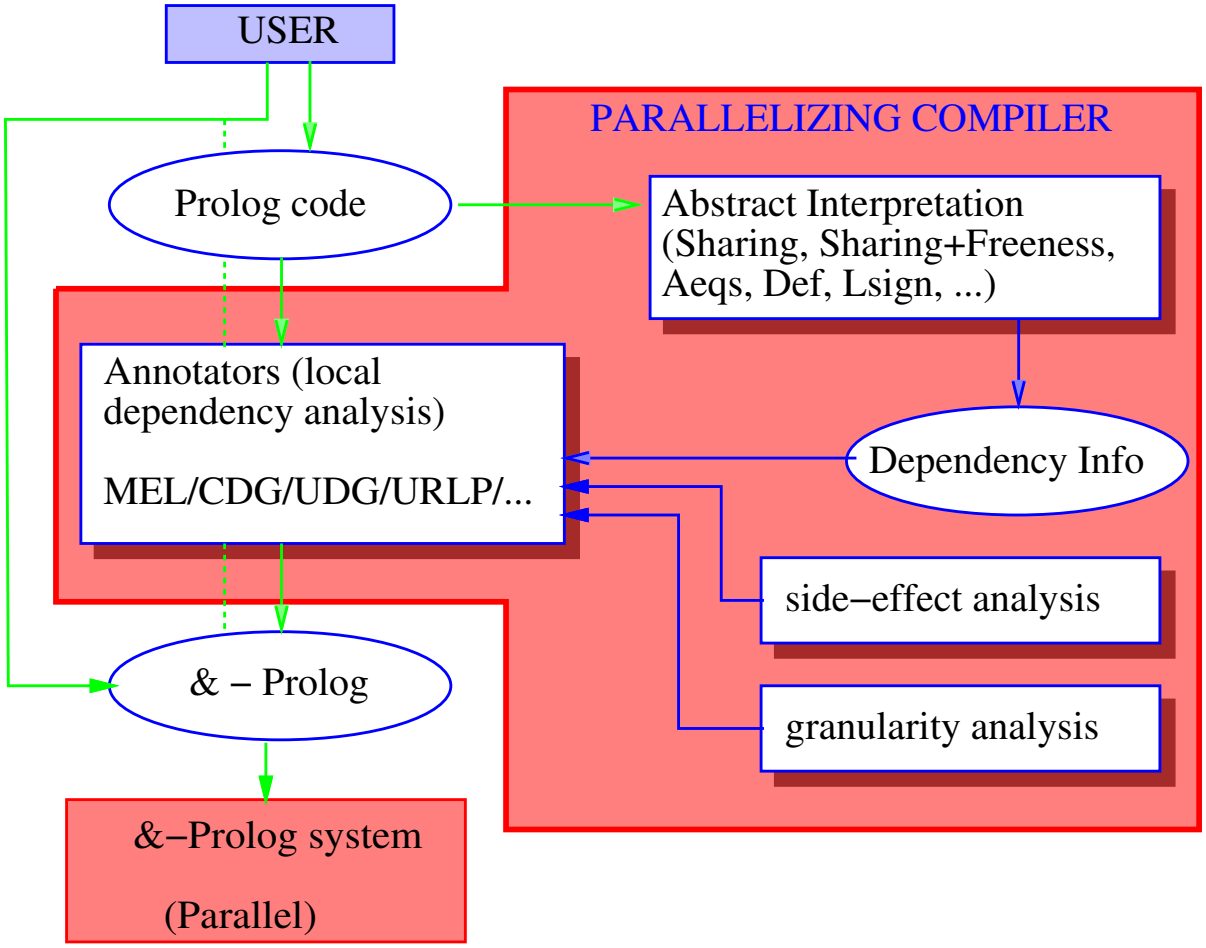
Parallelization Process

- Conditional dependency graph (of some segment, e.g., a clause):
  - vertices are possible tasks (statements, calls,...),
  - edges=possible dependency (labels=conditions needed for independence).
- Local or global analysis used to reduce/remove checks in the edges.

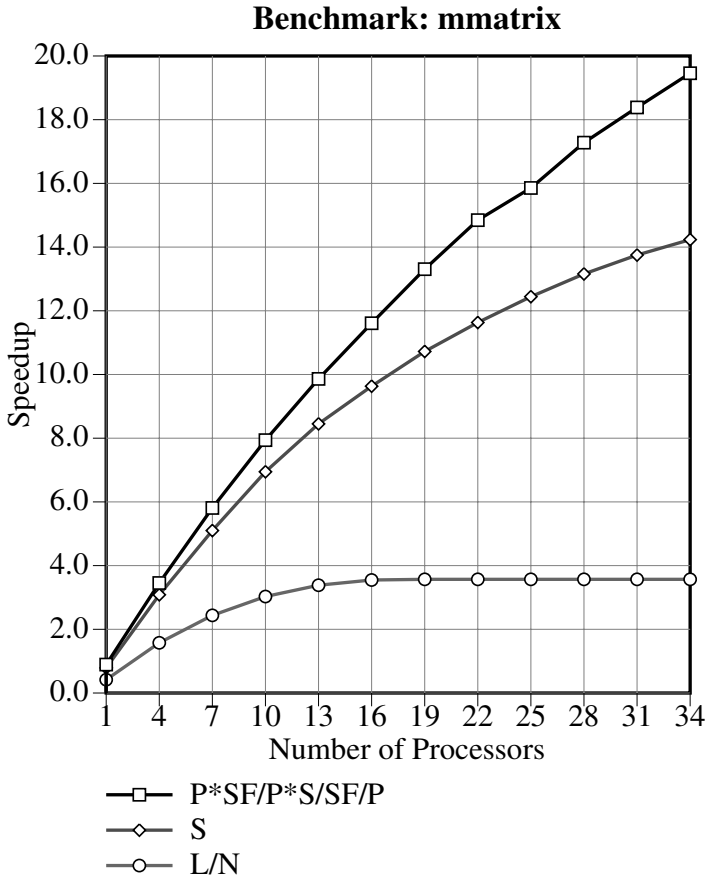
```
foo(...)  :-
     g1(...),
     g2(...),
     g3(...).
```

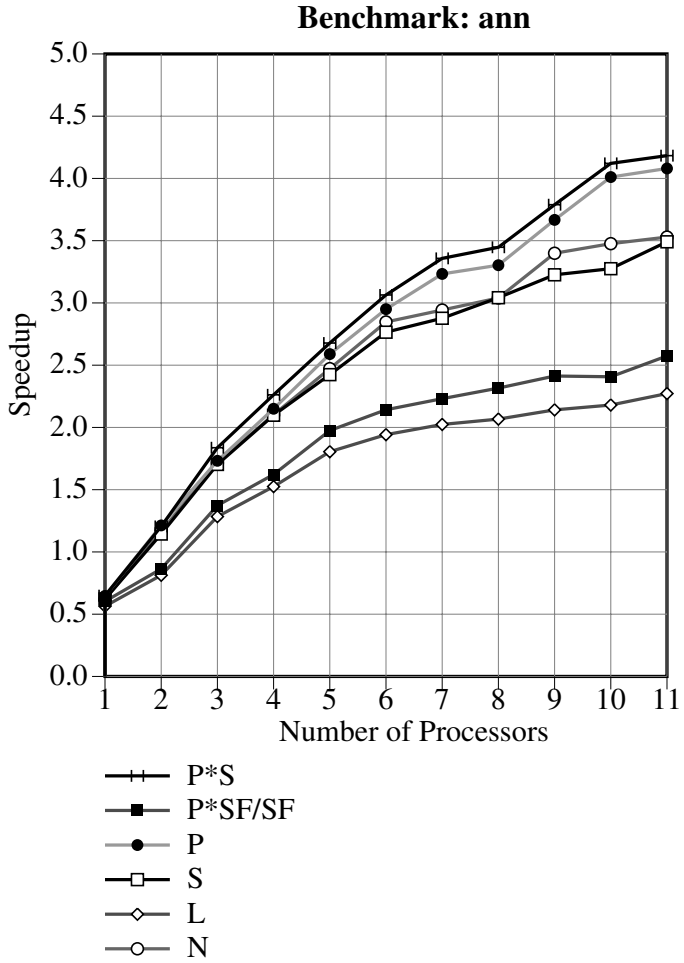


&-Prolog (and, later, CIAO) Parallelizer Overview –circa 1988



Some Speedups (Using Different Abstract Domains)



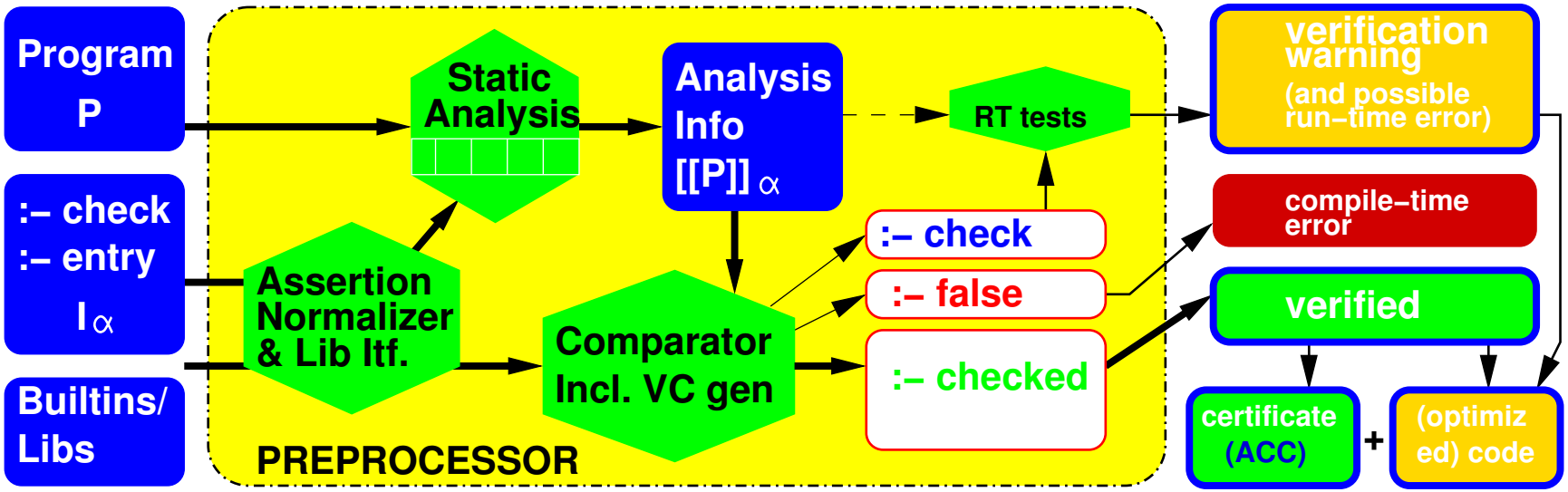


(ann is the parallelizer parallelizing itself;
1-10 proc.: actual speedups on Sequent Symmetry; 10+ simulator projections from execution traces)

The Second Phase: Program Development using AI

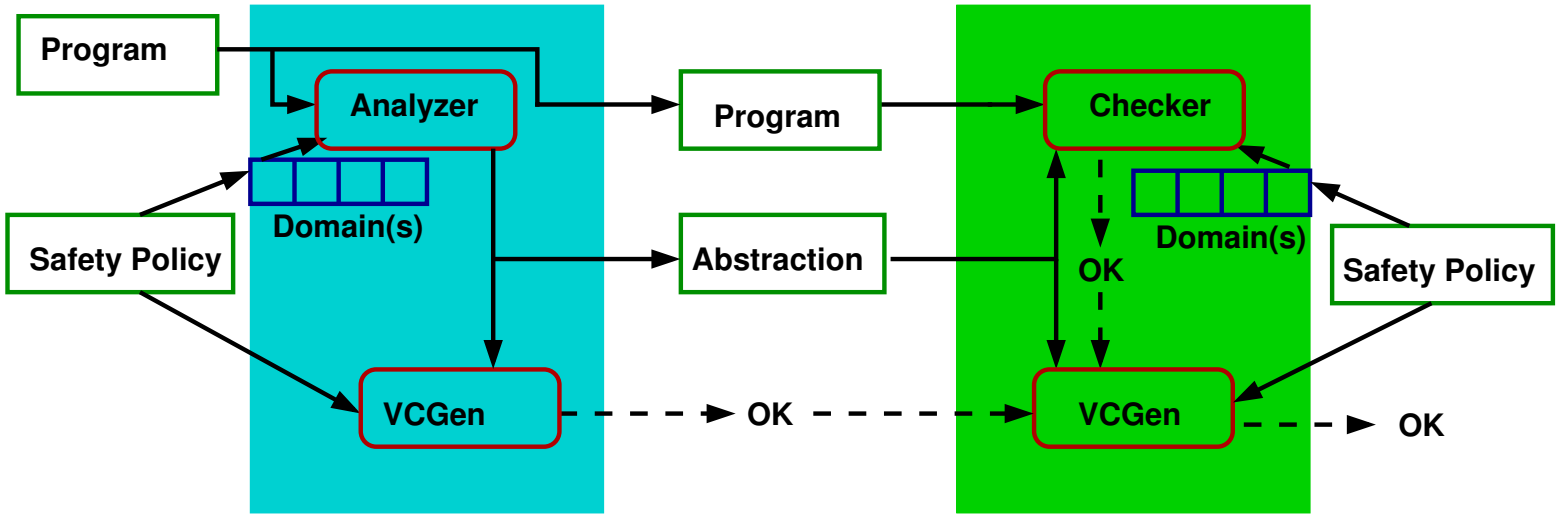
- The next stage:
  - not just optimization (parallelization, abstract partial evaluation, ...), but
  - **program development** fundamentally based on abstract interpretation.
- Ideas lurking for a long time, crystallized in first prototypes in 1994-97 [PPCP'94, Ciao system design], and:
  - Overall design [ILPS'97, AADEBUG'97, ICLP'99, LOPSTR'99, SAS'03, SCP'05]
  - Assertion language [LNCS 1870]
  - Modular extensions [Pietrzak, Correas, Puebla, Herme. LPAR'06]
  - Abstract diagnosis [Alpuente, Comini, Escobar, Falaschi, Lucas, LOPSTR'02] [Gallardo, Merino, Pimemtel, SAS'02] [Pietrzak, Herme. ICLP'07]
  - Extensions to Java [Albert, Gómez-Zamalloa, Hubert, Puebla PADL'07]
- Great influence of AI in **programming language design**:
  the "Ciao" language design is based on an *AI-based* compiler / env.
  (assertion based vs. strongly-typed, type systems as domains, etc.).
- Also, led us to the idea of Abstraction Carrying Code (ACC).

The CiaoPP Verification/Diagnosis Framework [AADEBUG'97]



- I_α (partial specification) provided via a language of *optional* assertions.
  - State properties at relevant points.
  - Talk about properties which can be predefined or user-defined.
    Types, cost, data sizes, termination, pointer aliasing, no-except, ...
- Implemented for Ciao and (less mature) for Java and Java bytecode.

The Abstraction Carrying Code (ACC) Scheme



$[\![P]\!]_\alpha = \text{Analysis} = \text{lfp}(analysis\_step)$ | Certificate $\subset [\![P]\!]_\alpha$ | Checker $= analysis\_step$

- Basic proposal [Albert, Puebla, Herme. COCV'04, LPAR'04, ICLP'04, PPDP'05, NGC'08]
- Incremental version [Albert, Arenas, Puebla LPAR'06]
- Certificate reduction (fixpoint compression) [Albert, Arenas, Puebla, Herme. ICLP'06]
- MOBIUS project [TGC'06].
- Incorporated in `CiaoPP` for types, modes, shapes, sizes, cost, det, termination, non-failure, resources, etc.

Combination with Partial Evaluation; Optimization; Scalability

- Combination with Partial Evaluation:
  - Abstract executability. Multiple abstract specialization [Puebla, Herme. PEPM'95]; application in parallelization [LOPSTR'97, JLP'99]
  - Full integration of Partial Evaluation and AI [Puebla, Albert, Herme. SAS'06]

  $\longrightarrow$ important contribution to the AI-based program development model.
- Application to program optimization (other than parallelization): abstract machines and native compilation [Morales, Carro, Herme. PADL'04, ICLP'05, LOPSTR'06]; embedded systems [Carro, Morales, Muller, Puebla, Herme. CASES'06].
- Modularity, Scalability, Practicality Issues:
  - Efficient, incremental intermodular analysis / fixpoint calculation [Bueno, G. de la Banda, Herme., Marriott, Puebla, Stuckey, Correas LOPSTR'01, LOPSTR'04, LOPSTR'05].
  - Widening of set-sharing [Navas, Bueno, Herme. PADL'06]
  - Dealing with the full ISO-Prolog standard [Bueno, Cabeza, Herme., Puebla ESOP'96]
- Domain combinations, goal-dependent vs. goal indep. analysis (flow sensitivity) [Codish, Mulkers, Bruy., G. de la Banda, Herme. PEPM'93, LPAR'94, TOPLAS'95, JLP'97]

Inference Resource-related and other Complex Properties

- Cost, resources (motivated by granularity control –lots of AI!):
  - Upper bounds (execution steps); application to granularity control [Debray, Lin, Herme., PLDI'90].
  - Lower bounds (execution steps) [w/López-García, ILPS'97, SAS'94].
  - Extension to Java bytecode [Albert, Arenas, Genaim, Puebla, Zanardini ESOP'07].
  - Execution times [Mera, López-García, Puebla, Carro, Herme., PADL'07].
  - User-definable resources for LP [Navas, Mera, López-García, Herme., ICLP'07].
  - Heap space analysis (Java bytecode) [Albert, Genaim, Gómez-Zamalloa, ISMM '07].
- Non-failure / no exceptions [Bueno, Debray, López-García, Herme. ICLP'97]; multivariant version [FLOPS'04].
- Determinacy analysis [Bueno, López-García, Herme. LOPSTR'04].
- Heap / shape analysis (C++/Java) [Marron, Herme., Kapur, Stefanovic LCPC'06, PASTE'07, CC'08]; Type domains, type widenings [Vaucheret, Bueno SAS'02].
- Termination analysis (Java) [Albert, Arenas, Codish, Genaim, Puebla, Zanardini WST'07].

Some Other Notable Facts

- Numerous PhD. theses:
  - 7 finished (C. Ochoa, D. Cabeza. P. López, G. Puebla, F. Bueno, M. G. de la Banda, K. Muthukumar). and 6 more to be finished in 2008/2009 (J. Correas, E. Mera, J.F. Morales, J. Navas, M. Méndez, A. Casas).

  + Many MS theses.
- Invited talks/keynotes, and tutorials:
  - On Abstract Interpretation: ICLP'92, WSA'92.
  - On Parallelism: EUROPAR'97, EUROPAR'04, DAMP'06, DAMP'07.
  - On Verification/Debugging using AI: AADEBUG'97, ILPS'97, ICLP'99, CL'00, ICALP'02, SAS'03, ACM SAC'06.
  - On Cost and Resource Analysis: SAS'94, PPDP'05.
  - On Abstraction Carrying Code: MoveLog'05, EAAI'06, TGC'06.
  - On the Combination w/Partial Evaluation: PEPM'03.
- Summer schools: LP'90, ESLLI'96, NMSU'99, ...
- Many national and EU projects (but no EU network!), even several babies...

Final Thoughts / The Next 30/22 Years!

- Go mainstream, make it an everyday tool! –coll. w/Industry (e.g., ES_PASS).
- Further work on scalability, modularity, domains, widenings, ...
- A big push to parallelization.
- Improve diagnosis.
- Emphasis on resource-related properties; specially *user-defined* (much current work on this).
- Develop ACC further, apply in practice to, e.g., small devices (MOBIUS).
- Develop new, even more dynamic high-level languages, getting correctness and real speed (without cheating), thanks to AI.
- Come work at the IMDEA-Software Development Technology Institute in Madrid!

But mainly, thanks Radhia and Patrick for opening such a wonderful space for us to do research –we look forward to the next 30 together!