Picat: A Scalable Logic-based Language and System (Invited talk)

Neng-Fa Zhou

Brooklyn College, The City University of New York
2900 Bedford Avenue, Brooklyn, New York, USA
`zhou@sci.brooklyn.cuny.edu`

Abstract

This talk will give the design principles of the Picat language (`http://www.picat-lang.org`), highlight the high-level and intuitive abstractions provided by Picat for easy programming, and contemplate why Picat is more robust and scalable than Prolog and could be more accessible than Prolog to ordinary programmers for scripting and modeling tasks.

1998 ACM Subject Classification D.1.6 Logic Programming

Keywords and phrases Functions, Relations, Loops, Constraints, Tabling

Digital Object Identifier 10.4230/OASIcs.SLATE.2013.5

Despite the elegant concepts, new extensions (e.g., tabling and constraints), and successful applications (e.g., knowledge engineering, NLP, and search problems), Prolog has a bad reputation for being old and difficult. Many ordinary programmers find the implicit non-directionality and non-determinism of Prolog to be hard to follow, and the non-logical features, such as cuts and dynamic predicates, are prone to misuses, leading to absurd codes. The lack of language constructs (e.g., loops) and libraries for programming everyday things is also considered a big weakness of Prolog. The backward compatibility requirement has made it hopeless to remedy the language issues in current Prolog systems, and there are urgent calls for a new language.

Several successors of Prolog have been designed, including Mercury, Erlang, Oz, and Curry. The requirement of many kinds of declarations in Mercury has made the language difficult to use; Erlang's abandonment of non-determinism in favor of concurrency has made the language unsuited for many applications despite its success in the telecom industry; Oz has never attained the popularity that the designers sought, probably due to its unfamiliar syntax and implicit laziness; Curry is considered too close to Haskell. All of these successors were designed in the 1990s, and now the time is ripe for a new logic-based language.

Picat aims to be a simple, and yet powerful, logic-based programming language for a variety of applications. Picat incorporates many declarative language features for better productivity of software development, including explicit non-determinism, explicit unification, functions, constraints, and tabling. Picat lacks Prolog's non-logical features, such as the cut operator and dynamic predicates, making Picat more reliable than Prolog. Picat also provides imperative language constructs for programming everyday things. The resulting system will be used for not only symbolic computations, which is a traditional application domain of declarative languages, but also for scripting and modeling tasks.

Picat is a general-purpose language that incorporates features from logic programming, functional programming, and scripting languages. The letters in the name summarize Picat's features:

- **P**attern-matching: A *predicate* defines a relation, and can have zero, one, or multiple answers. A *function* is a special kind of a predicate that always succeeds with *one* answer. Picat is a rule-based language. Predicates and functions are defined with pattern-matching rules.
- **I**mperative: Picat provides assignment and loop statements for programming everyday things. An assignable variable mimics multiple logic variables, each of which holds a value at a different stage of computation. Assignments are useful for computing aggregates and are used with the `foreach` loop for implementing list comprehensions.
- **C**onstraints: Picat supports constraint programming. Given a set of variables, each of which has a domain of possible values, and a set of constraints that limit the acceptable set of assignments of values to variables, the goal is to find an assignment of values to the variables that satisfies all of the constraints.
- **A**ctors: Actors are event-driven calls. Picat provides *action rules* for describing event-driven behaviors of actors. Events are posted through channels. An actor can be attached to a channel in order to watch and to process its events. Picat treats threads as channels, and allows the use of action rules to program concurrent threads.
- **T**abling: Tabling can be used to store the results of certain calculations in memory, allowing the program to do a quick table lookup instead of repeatedly calculating a value. As computer memory grows, tabling is becoming increasingly important for offering dynamic programming solutions for many problems.

Picat is more expressive than Prolog for scripting and modeling. With arrays, loops, and list comprehensions, it is not rare to find problems for which Picat requires an order of magnitude fewer lines of code to describe than Prolog. Picat is more scalable than Prolog. The use of pattern-matching rather than unification facilitates indexing of rules. Picat is more reliable than Prolog. In addition to explicit non-determinism, explicit unification, and a simple static module system, the lack of cuts, dynamic predicates, and operator overloading also improve the reliability of the language. Picat is not as powerful as Prolog for metaprogramming and it's impossible to write a meta-interpreter for Picat in Picat itself. Nevertheless, this weakness can be remedied with library modules for implementing domain-specific languages.