POTASSCO SOLUTIONS®

Whitepaper 2025

Mobility & Digital Products

Spack: How a Logic Solver Tamed Supercomputer Software Complexity

Contact Information

info@potassco.com

www.potassco.com

Success Story

Clingo is a powerful Answer Set Programming (ASP) grounder and solver from the Potassco project, the team led by Prof. Dr. Torsten Schaub at the University of Potsdam.
Spack is the de-facto standard for package management within the high-performance Computing (HPC) community. The core of Spack is the concretizer, a dependency and build configuration solver that selects an optimal build configuration from a nearly infinite space of options. Clingo has been an invaluable tool for the Spack project since 2020, when they replaced a custom, incomplete concretizer written in Python with a new implementation written using Clingo. Clingo's expressiveness has enabled Spack to model complex dependency constraints and compose numerous optimization criteria into a single, coherent, maintainable solve – a feat that was intractable with their previous approach.

The Challenge

HPC software builds are frequently customized. Users may set custom versions, build options, compiler choices, compiler flags, or custom implementations of libraries like BLAS and MPI. The full size of this configuration space is astronomical. The Spack project was created to manage this complexity, but as its capabilities grew, so did the burden on its core dependency solver.

Spack's original concretizer was an ad-hoc, greedy algorithm written in Python. While functional for simple cases, it had critical flaws:

Incompleteness and Incorrectness

- The greedy approach often failed to find a valid software configuration even when one existed, or it would produce a suboptimal one. It couldn't backtrack effectively to explore the full solution space.

High Maintenance Burden

- As new features were requested, like support for different GPU architectures or optimizing to reuse existing installations, the solver's logic became increasingly complex and brittle. Every new constraint required intricate, imperative code, leading to bugs and making the system difficult to maintain and extend.

Inexpressible Logic

- Combining multiple user preferences and optimization goals (e.g., "use the newest versions," "reuse existing packages," "prefer certain compilers") was nearly impossible to implement correctly in the ad-hoc solver.

The Spack team realized that building and maintaining a correct, high-performance combinatorial solver was a monumental task in itself, distracting their primary goal of building a great package manager. A fundamentally new approach was needed.

Our Solution

After evaluating other technologies like SAT and SMT solvers, the Spack team chose to base its new concretizer on Clingo and the paradigm of Answer Set Programming (ASP). Instead of trying to write a complex search algorithm, the team could now declaratively state the rules of a valid software build and let Clingo handle the incredibly difficult task of finding a solution.

The team re-architected Spack's concretizer around Clingo. The new process involves:

Translating to ASP Facts

- Spack traverses its package recipes and user preferences, translating all the versions, dependencies, and constraints into a large set of ASP "facts."

A Declarative Logic Program

- A central logic program, written in ASP, defines what constitutes a valid, installable software stack. It includes rules for dependency resolution, feature selection, reuse of installed dependencies, version and microarchitecture compatibility, and "unification sets" that define groups of dependencies that need to enforce certain consistency constraints.

Composed Optimization

- Spack leveraged ASP's powerful optimization capabilities to encode over 20 different criteria into the solve. This has allowed Spack to find not just any solution, but the best one, by minimizing a series of weighted goals that reflect user intuition.

Solving with Clingo

- Composed Optimization , Spack leveraged ASP's powerful optimization capabilities to encode over 20 different criteria into the solve. This has allowed Spack to find not just any solution, but the best one, by minimizing a series of weighted goals that reflect user intuition.

By adopting Clingo, the Spack team offloaded the complex solver implementation to the experts at Potassco, allowing them to focus on expressing concisely the semantics of package management.

The Benefits

Integrating Clingo as the core of the concretizer was transformative for the Spack project:

Correctness and Completeness

- Finds Solutions Reliably: Clingo's complete search guarantees that if a valid configuration exists, Spack will find it, eliminating the frustrating failures of the old greedy solver.
- Optimal Choices: The ability to compose optimization criteria ensures users get the most intuitive result - for example, maximizing the reuse of already-installed packages while ensuring any newly built packages use the latest versions.

Reduced Complexity and Increased Velocity

- Declarative, not Imperative: New features that would have required hundreds of lines of complex Python can now often be implemented with a small number of declarative ASP rules.
- Accelerated Innovation: This simplification unlocked the ability to implement groundbreaking features that were previously considered too difficult, such as the "compilers as nodes" model, which treats compilers and their implementation- and language-dependent runtime libraries as resolvable dependencies in the graph. This has been a long-standing goal in the HPC community.

Enabled Advanced Features and Growth

- Explainable Failures: The team used ASP to build a powerful system for explaining why a solve fails, tracing conflicts back to their source - a feature that is critical for user experience but very difficult to build with an ad-hoc solver.
- Composable Criteria: The switch to Clingo is not simply a rewrite. ASP's declarative nature makes it extensible and composable in ways that the prior concretizer could never have supported. As new functionality is needed in Spack, new rules and semantics can in many cases be added without modifying existing logic. For example, the Spack team has prototyped integrating AI models of software compatibility [2] and has added support for ABI compatibility for binaries [3], neither of which required major rewrites to existing logic.

What This Means

The Spack story is a powerful testament to the value of building on foundational, general-purpose tools. By adopting Clingo, the Spack project demonstrated that a small, domain-focused team can solve world-class combinatorial problems without having to become experts in writing solvers. The declarative nature of Answer Set Programming provided a language to express the problem cleanly, while the robust, highperformance engine of Clingo provided the power to solve it. The Potassco suite, including Clingo, is available as open-source software at potassco.org, empowering other projects to follow Spack's lead and solve their own „impossible" problems.

References

- Todd Gamblin, Massimiliano Culpo, Gregory Becker, and Sergei Shudler. Using Answer Set Programming for HPC Dependency Solving. In Supercomputing 2022 (SC'22), Dallas, Texas, November 13-18 2022.
- Daniel Nichols, Harshitha Menon, Todd Gamblin, and Abhinav Bhatele. A Probabilistic Approach to Selecting Build Configurations in Package Managers. In Supercomputing 2024 (SC'24), Atlanta, GA, November 17-22 2024.
- John Gouwar, Gregory Becker, Tamara Dahlgren, Nathan Hanford, Arjun Guha, and Todd Gamblin. Bridging the Gap Between Binary and Source Based Package Management in Spack. In Supercomputing 2025 (SC'25), St. Louis, MO, November 16-21 2025. To appear.