TRANSFORMATION XCSP VERS CHR++

Encadrants : Vincent Barichard vincent.barichard@univ-angers.fr
Marc Legeay marc.legeay@univ-angers.fr
Univ Angers, LERIA, SFR MATHSTIC, F-49000 Angers, France

Financement : Projet CHRES **Période** : 7 avril - 18 juillet 2025

Contexte

CHR (Constraint Handling Rules) est un langage de règles à chaînage avant qui permet de définir des contraintes déclaratives au sens de la programmation logique par contraintes [2, 3]. C'est une extension de langage qui permet d'introduire des contraintes définies par l'utilisateur, c'est-à-dire des prédicats du premier ordre, dans un langage hôte donné comme `Prolog`, `Lisp`, `Java` ou `C/C++`. `CHR++`[1], une implémentation de CHR au-dessus de `C++`, est aujourd'hui utilisé dans plusieurs projets de recherche au LERIA. Les CSP (Constraint Satisfaction Problems) permettent de modéliser des problèmes de satisfaction de contraintes. Un CSP est défini par un ensemble de variables, chacune à valoir dans un domaine, liées entre elles par des contraintes. Une solution à un CSP est une affectation de toutes les variables satisfaisant la totalité des contraintes. Plusieurs formalismes existent pour décrire des CSP, notamment XCSP^3 (`https://www.xcsp.org/`), un langage de description d'un CSP basé sur XML. XCSP fournit à l'utilisateur un ensemble d'éléments de modélisation dont un large catalogue de contraintes prédéfinies prêtes à l'usage. Ce catalogue intègre des contraintes arithmétiques, ensemblistes, et un ensemble de contraintes globales comme `linear` et `alldiff`.

Transformer un CSP en programme CHR (exécutable avec `CHR++`) implique d'écrire toutes les règles CHR garantissant qu'une solution du programme CHR sera également une solution du CSP. Une façon d'aboutir à ce résultat serait de traduire les contraintes du CSP en entrée en un ensemble de règles CHR réalisant des actions de filtrage analogues. Il serait alors possible de traduire intégralement un modèle CSP en programme CHR qui calculerait la solution au problème posé.

Sujet

Des chercheurs du LERIA ont initié le développement d'un solveur de contraintes arithmétiques au-dessus de `CHR++`. Le but du stage est de continuer le développement de ce solveur en réalisant un parseur permettant de traduire une instance au format XCSP vers un programme `CHR++`. Ce parseur facilitera l'adoption du nouveau solveur, car il rendra possible la réutilisation directe d'un ensemble de benchmarks existants mis à disposition par la communauté travaillant à l'élaboration du format XCSP. Il permettra également d'élargir le catalogue de contraintes mises à disposition par `CHR++`, lui permettant de modéliser plus facilement de nouveaux problèmes.

References

[1] Vincent Barichard. CHR++: An efficient CHR system in C++ with don't know non-determinism. *Expert Systems with Applications*, 238:121810, 2024.

[2] J. Jaffar and J.-L. Lassez. Constraint logic programming. In *Proceedings of the 14th Annual ACM Symposium on Principles of Programming Languages*, pages 111–119, 1987.

[3] J. Jaffar and M.J. Maher. Constraint logic programming: A survey. *Journal of Logic Programming*, 19/20:503–581, 1994.