ASPViz: Declarative Visualisation and Animation Using Answer Set Programming

Owen Cliffe, Marina De Vos, Martin Brain, and Julian Padget

Department of Computer Science,
University of Bath,
United Kingdom
{occ,mdv,mjb,jap}@cs.bath.ac.uk

Abstract. Answer set programming provides a powerful platform for model-based reasoning problems. The answer sets are solutions, but for many non-trivial problems post-processing is often necessary for human readability. In this paper we describe a method and a tool for visualising answer sets in which we exploit answer set programming itself to define how visualisations are constructed. An exciting potential application of our method is to assist in the debugging of answer set programs that, as a consequence of their declarative nature, are not amenable to traditional approaches: visual rendering of answer sets offers a way to help programmers spot false and missing solutions.

1 Introduction

Answer Set Programming (ASP) is a methodology for solving NP and NP-complete problems by representing the problem as a logic program under answer set semantics, such that the answer sets correspond to the solutions of the problem. Although the answer sets represent the solutions of the problem encoded by the program, they only consist of sets of atoms that are true (rendering the missing ones false). To understand these answer sets, one must interpret them in the context of the problem domain, which for simple cases can be done relatively easily (by inspection) but, for any non-trivial problem/domain answer set size is typically large and some post-processing is needed. In addition, post-processing often plays an important role in verifying and debugging answer set programs. There is a large body of literature relating to ASP, for in-depth coverage including commonly used syntax see [1].

In this paper, we introduce ASPViz, a tool that enables end-users and ASP programmers to visualise answer sets using the declarative nature of ASP itself to produce graphical representations of solutions. Visualisation of a given domain is achieved by the construction of a small answer set program that defines how elements of the problem solution should be displayed.

2 Declarative Visualisation with ASP

ASPViz: is a Java program that constructs two-dimensional images from the answer sets of a given program. The tool takes an answer set program $\Pi$, representing
a given problem and a visualisation program $\Pi_v$ which elaborates on the conclusions drawn by the program $\Pi$ concluding the necessary literals to render a graphic. Visualisation programs conclude atoms in the language $\mathcal{L}_{viz}$, whereof the atoms have the following types: (i) Those defining scene-control properties, such as the display extents, scene-wide transformations and animation orderings (see below), (ii) Those defining colours, brushes (line and fill properties), sprites (2D bitmapped graphics) and text properties which may be referenced in drawing atoms; e.g. the atoms: \texttt{brush(thick). brush\_color(thick,rgb(0,0,0)). brush\_width(thick,3).}

define a black line brush called thick which is three units thick, (iii) Primitive drawing atoms that relate to the rendering of graphical artifacts including lines, (filled) polygons, ellipses, curves, sprites, and text. Each of these predicates includes the relevant points (as terms) to position the corresponding artifact and drawing properties (brush, font, colour) required to render it; e.g. the atom \texttt{draw\_line(thick, p(0,0),p(1,4))} would draw a line connecting the points (0, 0) and (1, 4) using the defined brush thick.

ASPViz supports two visualisation modes: one frame, or multiple frames, per answer set. In the first case a visualisation program $\Pi_v$ is constructed such that it contains no negative-order cycles and is stratified below $\Pi$ (i.e. $\Pi_v$ is deterministic w.r.t. $\Pi$). The program $\Pi_v + \Pi$ is solved to give zero or more answer sets, each corresponding to an answer set of the original program $\Pi$ extended with visualisation atoms in $\mathcal{L}_{viz}$ concluded by $\Pi_v$. For each of these answer sets ASPViz extracts the atoms in $\mathcal{L}_{viz}$ and produces a graphic using a Java-native graphics toolkit (SWT) as follows: (i) Canvas extents and transformation are set using the defined properties—or default values where none are specified, (ii) Native objects (in SWT) are constructed corresponding to the defined colours, brushes, fonts and sprites, (iii) Each of the graphical primitives is rendered using the corresponding native drawing functions to produce a graphic.

A full description of language used, detailed examples and software are available from \url{http://www.cs.bath.ac.uk/~occ/aspviz/}

In addition to rendering answer sets of a program individually, the same approach may be used to create animations and multi-framed image visualisations based on individual answer sets. In this case the negative cycle restriction on $\Pi_v$ is relaxed yielding a program which may itself have multiple answer sets (a typical example might be a program which extracts each step of a plan as a frame). For each answer set of $\Pi$ a corresponding ground program $\Pi_i$ is produced consisting of the atoms of that answer set. $\Pi_v + \Pi_i$ is then solved yielding zero or more answer sets, each consisting of a partial rendering of the original answer set of $\Pi$. These frames are rendered as before. In order to produce animations, frames must be ordered: this is achieved by the inclusion of a unique atom \texttt{frame(T)} in the conclusions of $\Pi_v$, where T is a term. Animations are produced by parsing all produced answer sets of $\Pi_v + \Pi_i$ and then ordering the sets by the value of T. The value of T is typically an integer and may be determined directly from some value in $\Pi$ (e.g. the largest time step value in the case of of a planning problem), or may be derived from aggregates or weight values (where supported by the underlying ASP system) over atoms in a solution of $\Pi$. 