WORKCRAFT: framework for interpreted graphs

μSystems Group, School of Electrical and Electronic Engineering, Newcastle University, UK

A large number of models that are employed in the field of concurrent systems design, such as Petri nets, gate-level circuits, dataflow structures, etc. - all have an underlying static graph structure. Their semantics, however, is defined using additional entities, e.g. tokens to denote the overall state of the system. We jointly refer to such formalisms as *interpreted graph models*. The similarities in notation allow for links between different models to be created, such as interfaces between different formalisms or conversion from one model type into another, which greatly extends the range of applicable analysis techniques.

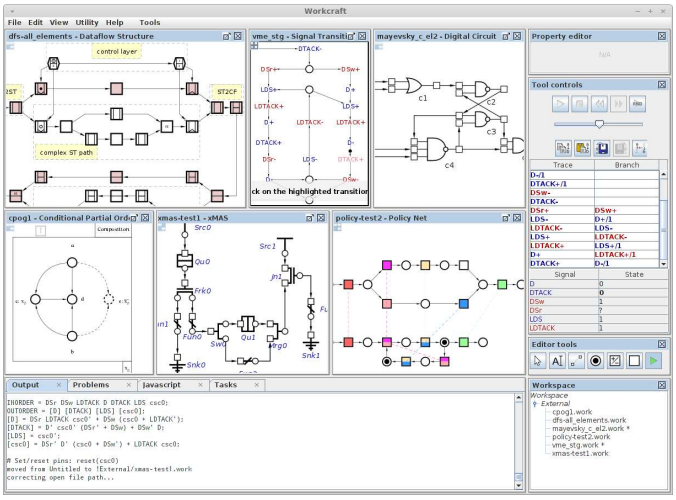

Figure 1. WORKCRAFT interface

WORKCRAFT is designed to provide a flexible common framework for development of interpreted graph models, including visual editing, (co-)simulation and analysis. The framework provides a graphical frontend for convenient manipulating the interpreted graph models (see Figure 1) and relies on a number of established backend tools for model checking, synthesis, and verification tasks (PETRIFY, PUNF, MPSAT, etc). The simulation and analysis of the models can be carried out either directly or by mapping a model into a behaviourally equivalent model of a different type (usually a Petri net). Hence the user can design a system using the most appropriate formalism, or even different formalisms for the subsystems, while still utilising the power of Petri net analysis techniques.

The framework is platform-independent, highly customisable by means of plugins, and is freely available for academic use at *http://workcraft.org/*. Currently the following interpreted graph models are supported by Workcraft plugins (see Figure 2 for model taxonomy and available conversion mechanisms):

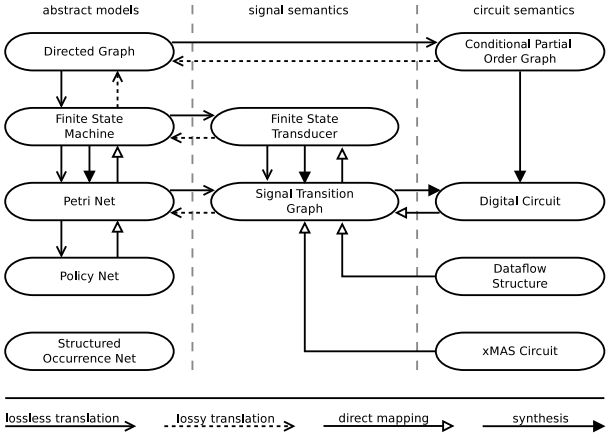

Figure 2. Supported graph models

- Directed Graph – a basic graph model that serves a basis for other interpreted graph models.
- Finite State Machine – a model for explicitly representing the state space of concurrent systems.
- Petri Net – a model for capturing causality and concurrency relations between asynchronous events.
- Policy Net – an extension of Petri nets with step execution semantics, convenient for modelling GALS systems [1].
- Structured Occurrence Net – a model for capturing and analysing families of system execution traces [2].
- Finite State Transducer – a Finite State Machine whose arcs have semantics of digital signal events.
- Signal Transition Graph – a labelled Petri net whose transitions are associated with signal events [3].
- Conditional Partial Order Graph – a formalism for specifying a collection of behavioural scenarios in a compact graph representation using optimal encodings [4].
- Digital Circuit – a gate-level model and verification infrastructure for asynchronous control circuits [5].
- Dataflow Structure – an asynchronous pipeline model that supports counterflow and spread token semantics [6].
- xMAS Circuit – a high-level microarchitectural model of communication fabrics extended to GALS systems [7].

REFERENCES

[1] J. Fernandes, et al: *"Persistent and nonviolent steps and the design of GALS systems"*, Fundamenta Inf., 2015.
[2] M. Koutny, B. Randell: *"Structured occurrence nets: a formalism for aiding system failure prevention and analysis techniques"*, Fundamenta Inf., 2009.
[3] A. Yakovlev, et al: *"A unified signal transition graph model for asynchronous control circuit synthesis"*, Formal Methods in System Design, 1996.
[4] A. Mokhov, A. Yakovlev: *"Conditional partial order graphs: model, synthesis and application"*, IEEE Trans. Computers, 2010.
[5] I. Poliakov, et al: *"Automated verification of asynchronous circuits using circuit Petri nets"*, ASYNC, 2008.
[6] D.Sokolov, et al: "Analysis of static data flow structures", Fundamenta Inf., 2008.
[7] F. Burns, et al: *"GALS synthesis and verification for xMAS models"*, DATE, 2015.