CDL — A Language for Cellular Processing

Christian Hochberger (hochberg@informatik.th-darmstadt.de)

Fachgebiet Mikroprogrammierung, Institut für Systemarchitektur,
Technische Hochschule Darmstadt
Alexanderstraße 10, D-64283 Darmstadt, Germany

1 Why Another Language?

Our goal is to describe complex cellular automata in a concise and readable way. The description of complex cellular automata models requires two supporting language elements:

1. The cell state must be a record of arbitrary types like enumerations, integer, subranges of both and maybe even floating popint values. It might also be desirable to have unions, to reduce the memory complexity of the cell state.
2. Testing of complex conditions requires deeply nested *if–then–else* or *case* statements. Support for complex automata must include special language elements that simplify the testing of complex conditions.

The descriptions shall be usable on arbitrary target architectures including special hardware simulators. Classical programming languages and already existing cellular programming languages fail to overcome these constraints, thus we created the language CDL. Some references to cellular programming languages: Cellang[1], CAL[2], Ceprol[3]

2 Features of CDL

Our language has the following features:

- The cell state can be a record or union of arbitrary types. The basic types supported are: enumeration, integer, float (may not be available in all implementations) and the special type `celladdress`.
- Through enumerations the CDL provides symbolic state names. The compiler does not necessarily have to encode the symbolic state names as they are ordered in the enumeration. This gives the compiler the opportunity to optimize the code.
- Special statements (`all, one, num`) are provided, which make the testing of complex conditions easy.
- In CDL all loops are unrollable at compile time. This makes it easy to translate the language into combinational logic.
- CDL contains a very simple yet powerful visualization concept. A colour can be assigned to cells fulfilling certain conditions.

3 Target Architectures for CDL

Three target architectures are currently supported by CDL:

1. A software simulator, which is based on X11 and Unix. Thus it is very portable. The CDL compiler translates the rule description into a C–function which is then compiled by the system compiler. Additionally the compiler generates control information for the visualizer. Details on the transformation process can be found at [4].

   This simulator has several drawbacks, so we are in the process of reimplementing it.

2. The CEPRA8–L machine[5]. This is a specialized hardware for the simulation of cellular automata. It is capable of simulating approximately 5 million cells per second. The memory complexity of a cell is limited to eight bit. The visualization of the entire field is done concurrently with the computation of the next generation. The next state is evaluated through a FPGA. Eight FPGAs work in parallel.

   The CDL compiler generates logic equations from the description. These equations are then processed by a set of commercial tools. The resulting design data can be downloaded into the FPGAs to load a new rule. For further information [6] may be referenced.

3. The CEPRA1–L machine. It uses a FPGA to compute the next state like the CEPRA8–L machine. The cellular field of the CEPRA1–L is stored in the host systems memory. This allows arbitrary sized fields, but implies the usage of the host systems graphic adaptor for the visualization.

4 Future Enhancements

There are several plans to enhance CDL:

- CDL will support multiple rules that are based on the same cell structure. This is especially useful for simulators which try to optimize the computation of the field. More details on this subject will be presented at the workshop by R. Schneider.
- CDL will be enhanced to support more complex visualization concepts, e.g. building patterns from cell clusters.
- CDL will include new elements to describe moving objects. The user will only need to specify the direction for an object and how conflicts shall be resolved, if objects would collide.

References

[1] Eckart, J.D.: A Cellular Automata Simulation System, SIGPLAN Notices **26** (August 1991) 80–85

[2] Palmer, I. J.: Scamper, available by anonymous ftp from ftp.uu.net.

[3] Seutter, F.: CEPROL – Eine zellulare Programmiersprache, Dissertation TU–Braunschweig 1985

[4] Hochberger, C., Hoffmann, R., Waldschmidt, S.: Compilation of CDL for Different Target Architectures, In Parallel Computing Technologies (LNCS 964), Springer Verlag, Editor: Malyshkin, V.

[5] Hoffmann, R., Völkmann, K.-P., Sobolewski, M.: The Cellular Processing Machine CEPRA-8L, in Parcella 94,179-188, Mathematical Research Vol. 81, Akademie Verlag 1994 Editors: Jesshope, Jossifov, Wilhelmi

[6] Waldschmidt, S., Hochberger, C.: FPGA Synthesis for Cellular Processing, In Proceedings of the 1995 IEEE/ACM International Workshop on Logic Synthesis