A CEP Technology Stack for Situation Recognition on the Gumstix Embedded Controller

Stephan Grimm, Thomas Hubauer, Thomas Runkler
Siemens AG, Corporate Technology, Munich, Germany

Carlos Pachajoa, Felix Rempe, Marco Seravalli, Philipp Neumann
Technische Universität München, Munich, Germany

Abstract: Semantic technologies – especially for symbolic reasoning and complex event processing – are particularly interesting to be employed on embedded controllers for various industrial applications such as diagnostics of technical devices to reason about sensor events. However, these technologies are typically tailored towards common PC infrastructure and are thus not readily available on embedded platforms. In this paper, we present a proof-of-concept implementation of a technology stack for semantic complex event processing on the Gumstix embedded platform and report on first experimental results about memory consumption.

1 Introduction

Semantic technologies based on symbolic reasoning are apt to be employed in various industrial use cases, such as knowledge-based diagnostics of technical machinery. There, diagnostic and other applications often need to be run on embedded devices that fit into an industrial environment and can e.g. be placed close to the sources of sensor signals. Hence, there is a need for getting semantic technology to run on embedded controllers.

However, available tools for knowledge-based reasoning are primarily designed for a PC infrastructure and can often not be easily run on embedded controllers with their restrictions on memory and CPU power and their often different architectures. It is not clear which of the available tools can be run or easily ported to particular embedded platforms.

A specific semantic technology suitable for the knowledge-based analysis of sensor signals is reasoning-enabled complex event processing (CEP), since sensor signal series are recorded over time and CEP-specific operators serve well the definition of respective rule-based recognition patterns for critical situations about technical machinery.

In this paper, we establish a specific technology stack for reasoning-enabled CEP based on Prolog and the ETALIS CEP-framework on the Gumstix[1] embedded controller, providing a basis for industrial situation recognition applications. In particular, we show the feasibility of the combination of the proposed technologies in a proof-of-concept implementation, report on lessons learned with the technologies involved as well as on experimental results about memory consumption, which is often the main obstacle for embedded reasoning.

[1] `www.gumstix.com`

2 Situation Recognition Patterns

In this section, we motivate the use of embedded CEP reasoning by means of industrial use cases, and we present some example patterns for situation recognition on sensor signals.

Industrial Use Cases for Situation Recognition

Industrial use cases for CEP-based situation recognition are often characterized by the lack of a PC-based computing infrastructure at the field control level of operating machinery. In areas such as factory automation or power generation, plants are typically operated by specific, dedicated control units (such as programmable logic controllers) not suitable to run CEP-based analysis next to the control tasks they perform. Moreover, plants are often operated in harsh environments exposed to heat, dust, vibrations and with restrictions on space for additional computing infrastructure for analysis and diagnostics. Hence, embedded general purpose computing platforms, such as the Gumstix system, are apt to be flexibly placed in field environments close to sensors and machine alarms and on top of pre-installed factory automation facilities. (In other cases, diagnostic reasoning is embedded into existing automation infrastructure as reported in [GWHC12], while CEP has also been considered for other specific devices in [DJHR+09, PLM12].) In the following, we list two example use cases that fit the above description.

Predictive Maintenance of Turbine Machinery based on Analysis of Sensor Time Series Unanticipated downtime of turbine machinery in electrical power plants typically goes along with high costs for power outages. Embedded devices can be used as intelligent sensors to constantly monitor a turbine's parameters based on sensor signals for temperature, vibration, etc., and to predict upcoming malfunctions based on on-board complex event processing on time-related patterns in sensor data prior to their occurrence. They are directly attached to critical turbine components to work in harsh power plant environments for an online prediction of critical operation states.

Diagnostics of Factory Automation Facilities based on Sensor Data and Alarm Event Logs In factory automation, malfunctions of production plants are typically documented by a multitude of alarm events logged by various plant control stations. In lack of a PC computing infrastructure at the factory control level, embedded devices can be used as flexible diagnostic units to analyze time correlations of alarm logs and sensor data using complex event processing directly at the field level for online diagnostics of automation equipment.

Furthermore, the CEP [EN10] paradigm combined with reasoning on semantic models is a suitable tool for the analysis of time-based correlations of field events and sensor time series in such use cases. The following is an example of a CEP-style rule in a simplified language that describes a situation where a temperature exceeds a threshold of 35 degrees constantly for a given time of 10 seconds. A (parameterized) sequence operator `SEQ` is used in combination with typical logic programming features, such as negation (NOT) and conjunction (AND), to realize time-related rule-based inference on events.

```
ContinuousThresholdExceedance(t)
  <-  SensorEvent (e1) AND e1.temperature > 35 AND
      SensorEvent (e2) AND e2.temperature > 35 AND
      e1 SEQ[10s] e2 AND t is e2.timestamp AND
      NOT (SensorEvent(e-) AND e-.temperature <= 35 AND e1 SEQ e- SEQ e2)
```

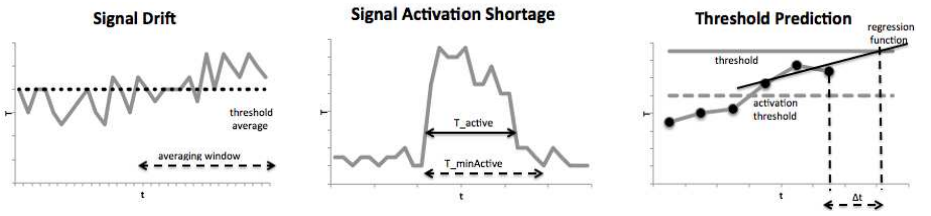


Figure 1: Patterns for situation recognition on sensor signals.

Situation Recognition Patterns

For the evaluation of our technology stack we used various recognition patterns that are typical for industrial use cases as the ones sketched above. They all define situations to be derived from particular arrangements of events using rule-based knowledge representation combined with the time-related operators of CEP. In industrial use cases they are typically applied to events that stem from sensor signals for temperature, vibration, pressure, etc. Some of these patterns are graphically illustrated in Figure 1 and listed in the following.

Signal Drift. When monitoring a device whose sensor signals are of noisy nature, it often is of interest to verify that the average value stays within a certain range despite the presence of single peaks. The pattern for recognizing signal drift sums up values in a given time window and checks whether their average value stays within a certain range - if not a drifting out of this range is being derived.

Signal Activation Shortage. For certain machinery (e.g. pumps, heatings) frequent changes between the On and Off state have a negative impact on their durability. Therefore, the time in the On or Off state shall be as long as possible. In order to detect too quick changes, this pattern monitors the duration of the active phase T_{active} and fires a warning, if it underruns a certain time $T_{minActive}$.

Threshold Exceedance Prediction. Machines should function within certain operational parameters. It would be useful to be able to determine if there is a possibility that a machine will step outside its safety region in the future. The pattern for threshold exceedance prediction estimates the future developing of a signal and derives a warning if there is a trend for surpassing a limit.

3 Technology Stack for Embedded CEP

As reported in [SS11], semantic technologies available for the PC world cannot always be easily run or ported to embedded platforms. In this section, we describe a working stack of semantic technologies for reasoning enabled CEP on the Gumstix embedded controller.

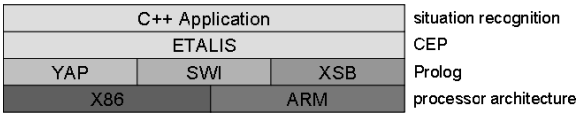


Figure 2: Software technology stack. (Layers can show alternatives for technologies.)

3.1 Embedded Controller Target Platform

The target platform chosen for this work is the Gumstix embedded controller, a very small general purpose computer made for ubiquitous computing applications that runs a Linux operating system. The choice for the Gumstix is due to various advantages over other embedded platforms, such as rather strong computational power and memory capacity, compatibility with available Linux-based tools and software libraries, good documentation for software development and best practises of usage in a large user community.

We were using a particular variant of the Gumstix computer-on-module product, namely the Over Earth controller together with the Tobi Expansion board, which provides external interfaces such as USB, Ethernet or HDMI. Altogether, the dimensions of this configuration are 105mm by 40mm by 14mm and it can be operated in temperatures up to 75 C, which makes it suitable for many industrial embedded applications. The Overo Earth controller is equipped with an OMAP 3503 micro processor that operates at 600MHz in an ARM Cortex-A8 architecture. It comes with 512 MB onboard primary memory and another 512 MB onboard secondary NAND memory. For second memory extension it provides a slot for microSD cards on which we used up to 8 GB.

3.2 CEP Software Technology Stack for Situation Recognition

Based on the Linux-driven Gumstix platform, we established a stack of technologies to realize embedded situation recognition, as shown in Figure 2.

Processor Architecture. The bottom-most layer of target processor architecture foresees the cross-compilation of all software involved to run on an ARM-architecture that is compatible to the Gumstix platform. Alternatively, however, development of situation recognition applications can also be done on a PC based on an x86 processor by using QEMU[2] as an emulator tool for ARM, and then ported to the Gumstix later.

Prolog Layer. On top of the Linux system that runs on the Gumstix, a Prolog engine is used as a basis to perform rule based reasoning. The most suitable Prolog reasoner for this work turned out to be SWI Prolog[3] [WSTL12]. This choice was due to the clear documentation, active user community and suitable C interface that this software provides.

Other alternatives have also been taken into account, namely XSB[4] and YAP[5]. Unfortu-

[2] http://wiki.qemu.org/
[3] http://www.swi-prolog.org/index.html
[4] http://xsb.sourceforge.net/
[5] http://www.dcc.fc.up.pt/~vsc/Yap/

nately, these reasoners could not be easily ported to the Gumstix system. For XSB we encountered technical issues with cross-compilation to the ARM processor architecture, while for YAP there was some problem with accessing the provided C interface library.

CEP Layer. In order to be able to treat the time dimension explicitly in semantic recognition models and to include time dependencies in rule-based reasoning, CEP is layered on top of Prolog. As a software solution for this CEP layer the ETALIS [AFR+10] framework was chosen. It extends the Prolog rule language by adding time operators that allow for the corelation of events. ETALIS comes as a set of Prolog rules and it can thus in principle be run with different underlying Prolog engines, as indicated in Figure 2. This makes ETALIS rules, formulated in an extended Prolog language, highly portable to other systems where Prolog is available without the need for rewriting rules.

Situation Recognition Layer. The purpose of this layer is to merge all the technologies involved for realizing knowledge-based recognition applications. The main functionalities are reading and parsing of data input, loading of rule bases and configuring of ETALIS, streaming data to the Prolog reasoner for CEP-based reasoning on the ETALIS rules.
The implementation was performed using C++, because of the low memory footprint of the resulting software and the possibility of having direct control over the memory of the Gumstix. Moreover, the majority of the Prolog reasoners offers a C interface, such that this decision of the programming language also offers higher possibility of future extensions and integration with additional engines.

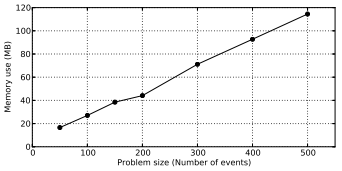
4 First Experimental Results

Evaluation of our implementation is based on the situation recognition patterns defined in Section 2. The number of events fed into the system is varied from 50 to 200. In order to test system behavior not only for increasing event load but also for an increasingly complex model, we also conducted experiments where the number of rules was artificially increased by multiplying the basic rule set (i.e. creating 1, 5, 10 or 15 copies of each rule).

In a first experiment, we analyzed memory consumption for an increasing number of events and a fixed rule set (one rule each for the various patterns). The results are depicted in Figure 3 a). The picture shows a clear linear dependency between the number of events processed and the memory required. This is an encouraging result, as especially memory is a highly valuable resource in embedded systems and any superlinear growth would therefore significantly reduce the application potential of our proposed solution.

In a second experiment, we tested system behavior for a growing number of rules at a fixed event count. More concretely, we created 1, 5, 10 or 15 copies of each rule and measured memory use for runs with 50, 100, 150 and 200 events each. Results of this experiment are shown in Figure 3 b). Our experiments reveal that (at least in the chosen settings where copied rules do not interact with each other), memory consumption also linearly depends on the number of rules. Moreover, increasing the number of events and rules seem to have a multiplicative effect on the total memory requirements. Again, linear growth in the number of rules is a positive result for the practical applicability of our solution.

a)



b)

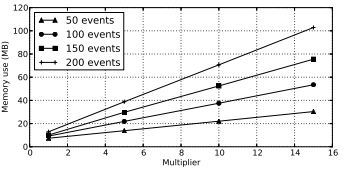


Figure 3: Memory use for a) increasing number of events, and b) multiplied rule bases.

5 Conclusions

In this paper, we have shown feasibility of a particular technology stack for embedded situation recognition based on semantic CEP and rule-based reasoning in a proof-of-concept implementation on the Gumstix embedded controller. We have reported on lessons learned with getting Prolog-based CEP to run on the Gumstix and pointed out various technology alternatives at certain layers. We have also reported on experimental results of reasonable memory consumption as one of the major constraints of typical embedded systems.

Our main finding here was that the linear development of memory use with increasing numbers of rules and events renders the proposed technology stack as appropriate for reasonable use cases. Hence, this work lays a basis for the realization of industrial ubiquitous computing applications based on reasoning-enabled CEP as a semantic technology.

References

[AFR+10] D. Anicic, P. Fodor, S. Rudolph, R. Stühmer, N. Stojanovic, and R. Studer. A Rule-Based Language for Complex Event Processing and Reasoning. In *RR*, volume 6333 of *Lecture Notes in Computer Science*, pages 42–57. Springer, 2010.

[DJHR+09] K. DoHyung, L. Jae-Ho, C. Ryu, C. Jung, and Y. Yong-Duck. Embedded CEP Engine Used in DDS-based Mobile Devices for Differentiated Services for Customers. In *IEEE 13th Int. Symposium on Consumer Electronics (ISCE).*, pages 645–646, 2009.

[EN10] O. Etzion and P. Niblett. *Event Processing in Action*. Manning Publications Co., Greenwich, CT, USA, 1st edition, 2010.

[GWHC12] S. Grimm, M. Watzke, T. Hubauer, and F. Cescolini. Embedded $\mathcal{EL}^+$ Reasoning on Programmable Logic Controllers. In *Proceedings of the 11th International Semantic Web Conference (ISWC'12)*, LNCS. Springer, 2012.

[PLM12] P. Pietrzak, P. Lindgren, and H. Makitaavola. Towards a Lightweight CEP Engine for Embedded Systems. In *IECON 2012 - 38th Annual Conference on IEEE Industrial Electronics Society*, pages 5805–5810, 2012.

[SS11] C. Seitz and R. Schönfelder. Rule-Based OWL Reasoning for Specific Embedded Devices. In *10th International Semantic Web Conference (ISWC)*, volume 7031 of *LNCS*, pages 237–252. Springer, 2011.

[WSTL12] J. Wielemaker, T. Schrijvers, M. Triska, and T. Lager. SWI-Prolog. *Theory and Practice of Logic Programming*, 12(1-2):67–96, 2012.