Cyclic Processing for Context Fusion

Mikkel Baun Kjærgaard*

Abstract. *Many machine-learning techniques use feedback information. However, current context fusion systems do not support this because they constrain processing to be structured as acyclic processing. This paper proposes a generalization which enables the use of cyclic processing in context fusion systems. A solution is proposed to the inherent problem of how to avoid uncontrollable looping during cyclic processing. The solution is based on finding cycles using graph-coloring and breaking cycles using time constraints.*

1. Introduction

Context information needed by context-aware applications is often not directly measurable by sensors. Therefore, sensor measurements need to be processed. This process of computing higher-level understanding from lower-level sensor measurements can (as in [2]) be defined as *"context fusion"*. The processing in terms of context fusion might take the form of estimation, where some specific information is sought. An example of this is the estimation of the location of a mobile device from a set of time measurements. To support context fusion, a specific type of system named a *context fusion system (CFS)* has been proposed. Different CFSs have been proposed within both the pervasive computing community, such as Solar [2], and the data management community, such as Aurora [1].

The role of a CFS is to interact with both sensors, context managers, and end-user applications. The interaction with sensors follows some sensor protocol, and context managers and end-user applications interact with the system using some context provision protocol. A context manager is an additional mediator between context information providers and end-user applications. The two main elements of a CFS are *operators* and *channels*[1]. The operators are implementations of individual processing steps. A single processing step could be, for instance, the application of a Bayesian estimator, a Kalman filter, an aggregation, or a mapping. The channels are data connectors between operators. Examples of different types of channels are *pull* channels, *push* channels and *periodic* channels. A problem with current CFSs is that they constrain processing to be structured as acyclic processing. In terms of channels and operators this means that combinations of these have to form acyclic graphs.

This paper proposes a novel generalization of such systems to allow combinations to form cyclic graphs. Context fusion with operators and channels forming cyclic graphs allows operators to use feedback information. The use of feedback information is a very common technique in machine learning. It is used, for instance, by priors in Bayesian estimation, Kalman filtering, and also often with hidden Markov models. The generalization to cyclic graphs allows such popular machine-

*Department of Computer Science, University of Aarhus, Denmark, email: mikkelbk@daimi.au.dk. The research reported in this paper was partially funded by the software part of the ISIS Katrinebjerg competency centre.

[1] Some systems use a different naming but have conceptually equal elements

learning techniques to be correctly modeled as combinations of channels and operators. However, a generalization to cyclic graphs needs to address the inherent problem of how to avoid uncontrollable looping when processing cyclic graphs. This paper proposes a solution to this problem based on time constraints on data processing. The proposed solution is consistent with how feedback information is used in popular machine-learning techniques such as the ones mentioned above.

However, CFSs also exist that do not directly represent the structure of processing, such as [3]. These CFSs do not constrain how operators are combined. This is because operators themselves have to look up the operators they depend on. Therefore, the operators can when programming be combined in anyway a developer would like. In comparison, the proposed solution enables cycles without a developer having to handle this when programming.

This paper focuses on sensors that produce continuous streams of data. From such streams, machine-learning techniques can estimate and detect relevant higher-level context information. Examples of such continuous streams are video camera feeds and streams of signal strength measurements. This is different from sensors, that are able to output discrete application events [5].

The paper is structured as follows. The proposed generalization of CFSs to cyclic graphs is further described in the next section. Then, the proposed solution for avoiding uncontrollable looping is presented in Section 3. A conclusion and a discussion of further work are given in Section 4.

2. Cyclic Graphs

Combinations of operators and channels can be modeled as a graph with channels as directed edges and operators as vertices. Figure 1 shows a combination of several channels and operators. The figure shows both an acyclic and a cyclic version of the same combination of channels and operators. The acyclic version has several problems:

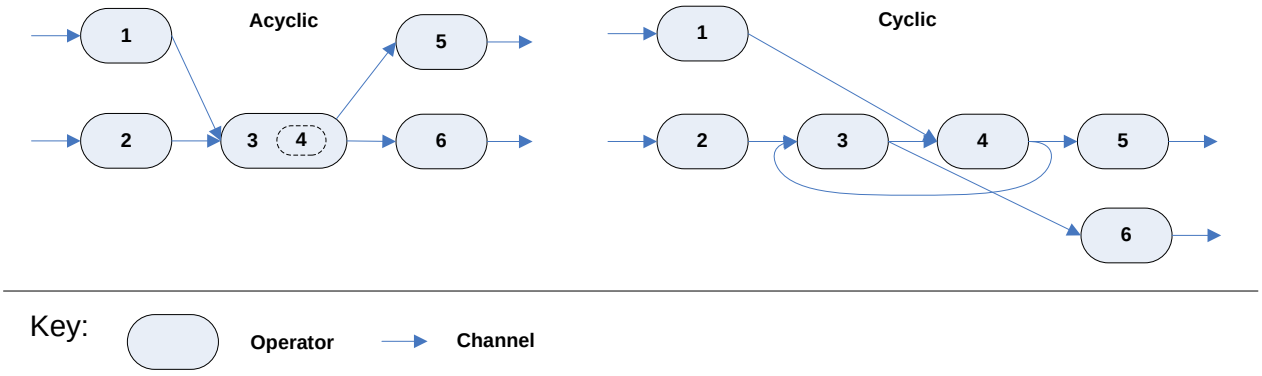

Figure 1. An acyclic and a cyclic operator graph

First, the acyclic graph does not allow the modeling of feedback information. For instance, if *operator 3* has to depend on feedback information from *operator 4*, the operators have to be combined in one operator, and the combined operator has to keep state to remember the last output of the combined operator. This means that an operator instance has to exist for each monitored entity. However, if cycles are used the last output information can be retrieved over the cycle as illustrated on the right in Figure 1. Therefore the cycles avoid that operators have to be combined and keep state to remember feedback information.

Second, the combination of operators to satisfy the acyclic constraint means that operators cannot be independently implemented and used. This is also illustrated on Figure 1. In the acyclic graph, illustrated on the left, *operator 3* and *operator 4* have to be composed for *operator 3* to use the output of *operator 4*. This means that *operator 4* cannot be independently changed and replicated. An additional problem is that *operator 3* also has to receive input from and provide output to operators which *operator 4* was connected to. These problems are completely avoided with cycles, as shown on the right in Figure 1. This means that operators can be implemented and used with a finer granularity.

These two benefits of cyclic graphs have several important implications for CFSs. First, because processing steps can be implemented statelessly, their implementations can be replicated on several devices, and can be moved between devices. Second, because of the finer granularity, single processing steps can now be changed and recombined in other setups to allow flexible processing. These benefits come with the cost of increased complexity of the CFS for handling the cyclic generalization.

3. Time Constraints

This paper proposes to apply time constraints to avoid uncontrollable looping during cyclic processing. These time constraints should be managed by the CFS. The time constraints should avoid uncontrolled looping by constraining processing of data sent over a *cycle-creating edge* which is the last edge to close a cycle. Figure 2, shows on the left an example of a combination of operators and channels including a cycle-creating edge illustrated as a dashed line. A CFS can automatically find the cycle-creating edges with a breadth-first coloring of all vertices. The breadth-first coloring starts from the vertices that represent operators which only have external inputs from sensors. When the cycle-creating edges have been found, the CFS can constrain the data sent over the channels represented by these edges. Channels should constrain data by delaying the point in time where data can be either picked up or forwarded. The time delays that should be enforced by channels depend on the frequency of data updates. Therefore, these delays should be calculated, for instance, from the frequency of sensor inputs or requests over the context provision protocol from context managers or applications.

In the following, a detailed example is given with a CFS having cyclic graphs, time constraints, push channels, and single-threaded execution of operators and channels. The single thread executes operators and channels in a order satisfying all non cycle-creating edges. Such an order can be determined by the same coloring algorithm as used above for finding the cycle-creating edges. The CFS calculates the frequency of execution of all operators and channels from requests over the used context provision protocol. Because the operators and channels are executed with a specific execution frequency we can count as time steps: $T_1, T_2, ...$ the executions of the operators and channels. A CFS matching these assumptions has been implemented as a prototype; however, the details of it are omitted from this paper.

On top of the described CFS, a *Location Fingerprinting (LF)* system has been realized following [4]. This LF system is based on Bayesian inference and use a Markov model for tracking. The operators and channels used to realize the LF system are shown in Figure 2. This Figure also illustrates the execution of the operators and channels for the LF system. The execution is illustrated for time step T_1 (the first time step) and $T_x, x > 1$ (any later time steps). In the first time step T_1, the *mapping operator* is executed which means that it picks up data on the ingoing push channel and outputs data on the outgoing push channel. The channels are marked T_1 to illustrate that this data is from time step T_1. The *Bayesian operator* is then executed and pushes data to the *aggregator operator*

and the *Markov operator*. The *Bayesian operator* is, however, not able to pick up any data from the *Markov operator*. The *Markov operator* is then executed and pushes data back to the *Bayesian operator*. Finally, the *aggregator operator* is executed and pushes data for output to applications or context managers requesting context information. In following time steps T_x, the operators are executed again in the same order. However, because the *Markov operator* outputted data in time step T_{x-1}, the *Bayesian estimator* will be able to collect data from it in time step T_x. As illustrated, data is delayed over the cycle-creating edge to enforce the time constraint. This delay is consistent with the Bayesian estimator's assumption about feedback information illustrating the solutions applicability with machine-learning techniques.

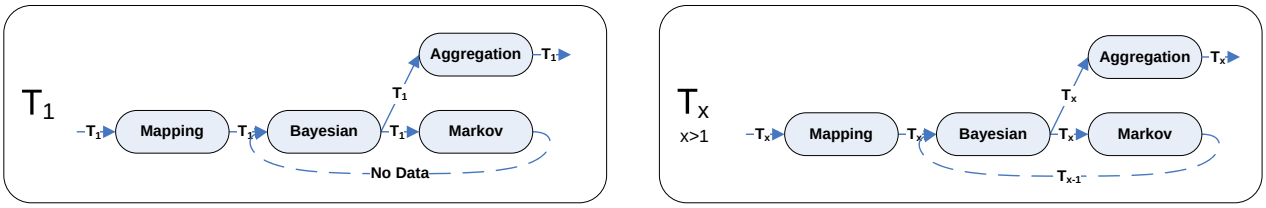

Figure 2. Execution of operators at time step T_1 and T_x for a LF system.

4. Conclusion and Further Work

This paper presented a generalization of CFSs to cyclic graphs. This generalization has several benefits in terms of stateless implementation, flexibility and replication. A solution was proposed to the inherent problem of how to avoid uncontrollable looping during cyclic processing based on enforcing time constraints on data. However, to further evaluate the idea it would be valuable to try out other kinds of machine-learning techniques and sensor inputs. Another relevant line of further work is to evaluate the extension of other CFSs with cyclic capabilities.

References

[1] D. Abadi, D. Carney, U. Cetintemel, M. Cherniack, C. Convey, S. Lee, M. Stonebraker, N. Tatbul, and S. Zdonik. Aurora: A New Model and Architecture for Data Stream Management. *VLDB Journal*, (12)2:120–139, August 2003.

[2] Guanling Chen, Ming Li, and David Kotz. Design and Implementation of a Large-Scale Context Fusion Network. In *Proceedings of the First Annual International Conference on Mobile and Ubiquitous Systems: Networks and Services*, 2004.

[3] Anind K. Dey, Daniel Salber, and Gregory D. Abowd. A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications. *Human-Computer Interaction (HCI) Journal*, 16 (2-4):97–166, 2001.

[4] Andreas Haeberlen, Eliot Flannery, Andrew M. Ladd, Algis Rudys, Dan S. Wallach, and Lydia E. Kavraki. Practical Robust Localization over Large-Scale 802.11 Wireless Networks. In *Proceedings of the Tenth ACM International Conference on Mobile Computing and Networking*, 2004.

[5] Samuel R. Madden, Michael J. Franklin, Joseph M. Hellerstein, and Wei Hong. TinyDB: An Acquisitional Query Processing System for Sensor Networks. *ACM Trans. Database Syst.*, 30(1):122–173, 2005.