On combining a context recognition system and a configuration planner for personalised ambient assisted living

Lia Susana d.C. Silva-Lopez, Jonas Ullberg, and Lars Karlsson

Center for Applied Autonomous Sensor Systems, Örebro University, SE-70182 Sweden
{lia.silva, jonas.ullberg, lars.karlsson}@oru.se

1 GiraffPlus system

This abstract describes how a context recognition system and a configuration planner can interact to enable personalised activity monitoring in apartments with different features. The two systems are developed to supporting independent living of senior citizens (primary users) by monitoring their daily interaction with the environment. For this purpose, networked non-intrusive sensors that senses motion, power usage, and pressure are used, in addition relevant physiological parameters such as heart rate and blood pressure are measured on demand. The system allows social interaction and timely involvement of family and caregivers, and is being developed within the European Commission FP7 project GiraffPlus [2]. The GiraffPlus system also contains a telepresence robot and an interface for caregivers and medical specialists (secondary users), to personalise and visualise data, alarms, and reminders for events of interest that have been permitted by the senior citizen. The environment is modelled as a collection of state variables that represent the layout of the apartment, position and motion of individuals and items (e.g. `person1` in `bedroom`, `book1` on `table` in `bedroom`), status of items in the apartment (e.g. `light` `on/off`, `door` `open/closed`), or physiological parameters (e.g. `heart-rate` of `person1`, `weight` of `person1`). Values of state variables are directly observable through sensors, or indirectly derivable from other state variables. There are also purely computational processes that refines sensed data to various levels of abstraction.

Now, consider an apartment with networked sensors (glucose, motion), actuators (alarms) and programs, as illustrated in the right hand side of Figure 1. Different apartments may be equipped with different sensors at different positions, have different layout and furniture, and all of the former can change position or be updated over time. In this way, the set of activities that are monitored and they way in which they are inferred varies between homes and can change over time. A configuration planner is being developed in order to enable activity monitoring under such varying conditions. The purpose of the configuration planner in the GiraffPlus project is to provide the context recognition with the data it requires for recognizing the activities requested by the users. When the physical setup of the apartment is represented in a domain along with the types of data the sensors handle, the effects of actuators, and the programs available for data processing, a configuration planner can be requested to obtain information or have desirable effects into the environment, by generating information flows from

Fig. 1. An example of GiraffPlus apartment. The figure shows a schematic view of interconnected devices and the location of sensors and a robot in the home.

sensors and sequences of activations of programs and actuators. The different components of the GiraffPlus system have been implemented and integrated, and tested in a laboratory setting. We are currently evaluating the GiraffPlus system in six apartments of elderly persons. The result of this evaluation, both in terms of input from users and in terms of technical functions, will be used for further refinement of the system in an incremental fashion, at later stages the system will be deployed in fifteen apartments in Italy, Spain and Sweden.

2 Configuration planner

The configuration planner allows separating descriptions of sensors and programs from descriptions of activities, which in turn allows the system to perform inferences in different apartments with different setups, in which elements are added, removed, or fail by *automatically* adapting to them. A **functionality instance** represents a program that either operates on a sensor or actuator, or processes data from and/or delivers data to other functionalities. In Figure 1, we represent devices by the information they sense (sensors in M1, M3, M5 and MRbt1), their actuating capacities (actuators in M2 and MRbt1), or by how they process information (computing elements may run programs). Functionalities are allowed to represent elements from different devices.

A **configuration** is a set of functionality instances to execute, a mapping of outputs of those functionalities to inputs of other functionalities in the same set, and mapping of causal dependencies and execution order constraints between them. Configurations involve activating elements to have an effect in the environment, providing programs with data from sensors, and even alternating combinations of performing effects and feeding programs with information. In a **fully admissible** configuration, no requirements of any functionality instance of the configuration are unsatisfied, and no effect threatens a condition for another functionality instance.

The representation for our planner is based on the work of Lundh et al. [7], and the planner is inspired by partial-order planning and more particularly UCPOP [11]).

Differences emerge from the interaction of information and causal requirements. In addition to single-task robots in [9, 3], our configuration planner is also capable of handling functionalities with more than one task, as long as the different tasks do not conflict with each other during execution. Moreover, our approach does not require the specification of hierarchical task networks as in [7], allowing every potentially useful source of information and causal effects to be considered when building a configuration, as opposed to only the ones foreseen when modelling the functionalities in a device. Finally, when coupled with an execution monitoring process, our approach can also be used for repairing failed configurations [6], since it searches in the space of partial configurations that are satisfied until admissibility, given a set of initial state variable assignments, a set of functionality instances, and the damaged configuration as a goal.

Our algorithm starts with an initial configuration to be refined until a solution is found. The initial configuration contains a functionality instance and its requirements as the requested goal, and initial execution order constraints imposed by the effects of a functionality instance in which the initial state is encoded, and by the goal functionality. The initial plan is refined by adding new functionalities, or using existing ones, in order to satisfy requirements of the present functionalities. The planner satisfies a causal requirement of a functionality f by the causal effect of another functionality f' that comes before in time, and satisfies an information requirements of f by the information output of functionality f' at the same time. Causal requirements and effect may interfere, and in such cases temporal constraints between the functionalities can be added in order to avoid that interference.

2.1 Tested heuristics

For guiding search, our planner can use one (or several) heuristics for pruning, a heuristic for ordering the search front according a criteria, and another heuristic to select a requirement to satisfy first. We refer to this combination of heuristics as a *heuristic chain*. The idea is inspired on combining value ordering (choosing the next node to satisfy) and variable ordering (choosing the next unsatisfied link to satisfy) heuristics in constraint satisfaction problems. So far we have studied local information heuristics that use a histogram built with the frequencies of the sources provided by every available functionality instance, and a histogram with the frequencies of the requirements of the partial configuration. Sources and requirements are associated in bins that share the same property and format, of separate histograms. We have also used the Rho heuristic [5], which branches into the children that maximize the solution density of a problem by choosing the requirement with most and/or the tightest constraints.

3 Context recognition

The context recognition system represents the states of the sensors and the inferred activities as intervals on different timelines. The model that describes the causal relationships between the states of the sensors and the inferred activities is provided as a set of quantitative Allen’s interval algebra [1] constraints that are posted between intervals representing sensor readings. For instance, the rule: `Cooking Equals Stove  $\wedge$`

Cooking During Kitchen (cooking implies the stove being on during presence in the kitchen) defines how an activity, in this case the activity Cooking, can be inferred from intervals representing sensed data. This approach to context recognition was first described by Ullberg et al. [12] and then subsequently extended by Pecora et al. [10]. However, in prior work it was found that, although constraints taken from Allen’s interval algebra provides a convenient way to describe relations, small deviations in how the raw sensory data is interpreted and placed on the timelines can prevent activities from being inferred and in some cases produce false positives. One possible way of overcoming this is by allowing constraint violations to some degree, for instance by using fuzzy Allen’s interval constraints [8]. However, this former approach requires thresholds on the likelihood of inferred activities, which is not always possible to determine.

The context recognition in the GiraffPlus project handles the problem by performing temporal inference on multiple intervals contextually, admitting several interpretations of sensor and activity timelines. That is, each sensor reading is represented as a set of flexible temporal intervals rather than one. The former approach to context recognition is described in [13] where it is compared to the Chronicle recognition approach to context recognition proposed by Dousson and Maigat [4], in which Allen’s interval constraints are also used.

The implementation of the context recognition system is implemented as three separate modules; preprocessing, inference and extraction. The preprocessing module fetches samples from the centralized data storage (which contains data from several homes) and generates a set of possible timeline candidates from them. The inference module takes these sets of timelines and uses temporal rules to propagate the constraints on the timelines, producing a set of possible activity timelines. Finally the extraction module inspects the set of activity timelines and extracts a single timeline that can be used by the rest of the system, for instance by visualizing the daily activities to a care-givers or to raise an automatic alarm.

4 Interactions between the context recognition and the configuration planner

The context recognition provides goals in the form of *entity . property . format* to refer to information to obtain in the form of a state variable, and when a *value* field is added to the request, the goal refers to changing the value of the state variable. For instance, if we wanted to ask the configuration planner to find a configuration in order to know in which room of the apartment a person is, we could do it with *person . location . rooms*. Similarly, if we wanted to move a robot the room in which the person is located, we could do it with *robot1 . location . rooms = person . location . rooms*, and if we wanted to move the robot to a particular position in the house, we could do it with *robot1 . location . coordinates = (33.5,44.2)*. The context recognition can add additional goals on the fly. After receiving a goal, the configuration planner searches in the space of possible partial configurations until a fully admissible configuration that satisfies the goal is found. When a configuration is found, functionalities can be executed with the help of an execution monitoring algorithm.

5 Ongoing work on incorporating preferences into the system

We are currently working on expanding the interaction of both systems so that they can support preferences when generating goals, when searching for ways of satisfying the goal, and when modelling the requirements of the functionality instances.

The preferences may come from the primary user, the secondary user, the context recognition, and the domain designers. Preferences coming from primary users usually involve permissions and express how comfortable or not is the senior citizen with certain ways of monitoring their health and activities, as well as for allowing certain services or not, and under which circumstances. Preferences coming from secondary users involve which activities and situations are of most importance for the system to keep track of, and which physiological signals too. Preferences coming from the context recognition can involve relations between how desirable are certain types of information sources in order to perform certain inferences, so richer goals can be formulated to maximize the chance of obtaining good information sources. Preferences coming from domain designers are used to describe how each program, sensor and actuator works on the system. For instance, preferences can be used to express the fact that a functionality that works as a thermostat will work as long as it has all information inputs with the temperatures of the rooms of the house, and it would *prefer* that all doors and windows are closed. If for receiving a guest a door needs to be open, that would still be fine and the configuration running a thermostat does not need to stop. It will be possible to state that the configuration may no longer be as reliable. However, if all thermometers fail, the configuration may stop, since the hard requirements ceased being satisfied.

Preferences will enrich resource handling: to sense blood sugar, shall we use data from the sensors a person is wearing, shall we ask the person to take a test on a glucometer that uses expensive stripes and also measures ketone concentrations but hurts the finger more, or shall we ask the person to use the simple glucometer with cheaper stripes?. Preferences will allow the planner to get more “adequate” configurations: the planner should avoid a configuration in which something uncomfortable for the senior citizen could happen, except perhaps for a life-or-death scenario. Preferences can also be used to express richer goals for the whole system: in the representation for a device, requirements can be expressed in terms of preferences e.g. a service can now state that it prefers to use the information gathered by the pressure sensor in the bed, rather than the information gathered by the movement sensor in the bedroom, in order to determine the amount of sleep that a person had. Preferences will be used to express reliability of functionalities according to the different ways in which they can be satisfied, which will be of help to have a better functioning and more dependable system e.g. it would make it possible to express that a thermostat functionality is more reliable if the condition “door in bedroom == closed” holds. Preferences can be used to express data quality as a function of how preferable are the sources that are being used to satisfy the requirements of a functionality instance. Our current challenge is to incorporate all of the former ways of working with preferences, and to be able to interact with the preferences stated by the context recognition and the users.

Bibliography

- [1] Allen, J.F.: Maintaining knowledge about temporal intervals. *Communications of the ACM* 26, 832–843 (Nov 1983), <http://doi.acm.org/10.1145/182.358434>
- [2] Coradeschi, S., Cesta, A., Cortellessa, G., Coraci, L., Gonzalez, J., Karlsson, L., Furfari, F., Loutfi, A., Orlandini, A., Palumbo, F., Pecora, F., von Rump, S., Stimec, A., Ullberg, J., Östlund, B.: Giraffplus: Combining social interaction and long term monitoring for promoting independent living. In: 6th International Conference on Human System Interactions (HSI) (2013)
- [3] Di Rocco, M., Pecora, F., Sivakumar, P.K., Saffiotti, A.: Configuration planning with multiple dynamic goals. In: Proceedings of the AAAI Spring Symposium on Designing Intelligent Robots, Stanford, California (2013)
- [4] Dousson, C., Maigat, P.L.: Chronicle recognition improvement using temporal focusing and hierarchization. In: Proceedings of the 20th international joint conference on Artificial intelligence. pp. 324–329. IJCAI’07, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (2007), <http://dl.acm.org/citation.cfm?id=1625275.1625326>
- [5] Gent, I., MacIntyre, E., Presser, P., Smith, B., Walsh, T.: An empirical study of dynamic variable ordering heuristics for the constraint satisfaction problem. In: Principles and Practice of Constraint Programming CP96. pp. 179–193. Springer (1996)
- [6] Gritti, M., Broxvall, M., Saffiotti, A.: Reactive self-configuration of an ecology of robots. In: Proceedings of the ICRA-07 Workshop on Network Robot Systems. (2007)
- [7] Lundh, R., Karlsson, L., Saffiotti, A.: Autonomous functional configuration of a network robot system. *Robotics and Autonomous Systems*, Elsevier 56(10), 819–830 (2008)
- [8] Mansouri, M.: Constraint-Based Activity Recognition with Uncertainty. Master’s thesis, Örebro University, School of Science and Technology (2011)
- [9] Parker, L.E., Tang, F.: Building multirobot coalitions through automated task solution synthesis. *Proceedings of the IEEE* 94(7), 1289–1305 (2006)
- [10] Pecora, F., Cirillo, M., Dell’Osa, F., Ullberg, J., Saffiotti, A.: A Constraint-Based Approach for Proactive, Context-Aware Human Support. *Journal of Ambient Intelligence and Smart Environments* (2012), (accepted)
- [11] Penberthy, J., Weld, D., et al.: Ucpop: A sound, complete, partial order planner for adl. In: proceedings of the third international conference on knowledge representation and reasoning. pp. 103–114. Citeseer (1992)
- [12] Ullberg, J., Loutfi, A., Pecora, F.: Towards Continuous Activity Monitoring with Temporal Constraints. In: Proceedings of the 4th Workshop on Planning and Plan Execution for Real-World Systems at ICAPS09 (2009)
- [13] Ullberg, J., Pecora, F.: Propagating temporal constraints on sets of intervals. *ICAPS Workshop on Planning and Scheduling with Timelines (PSTL)* (2012), files/papers/UllbergPecora-ICAPS12-WS-PSTL_v2.pdf