Representing Swarm Behaviors

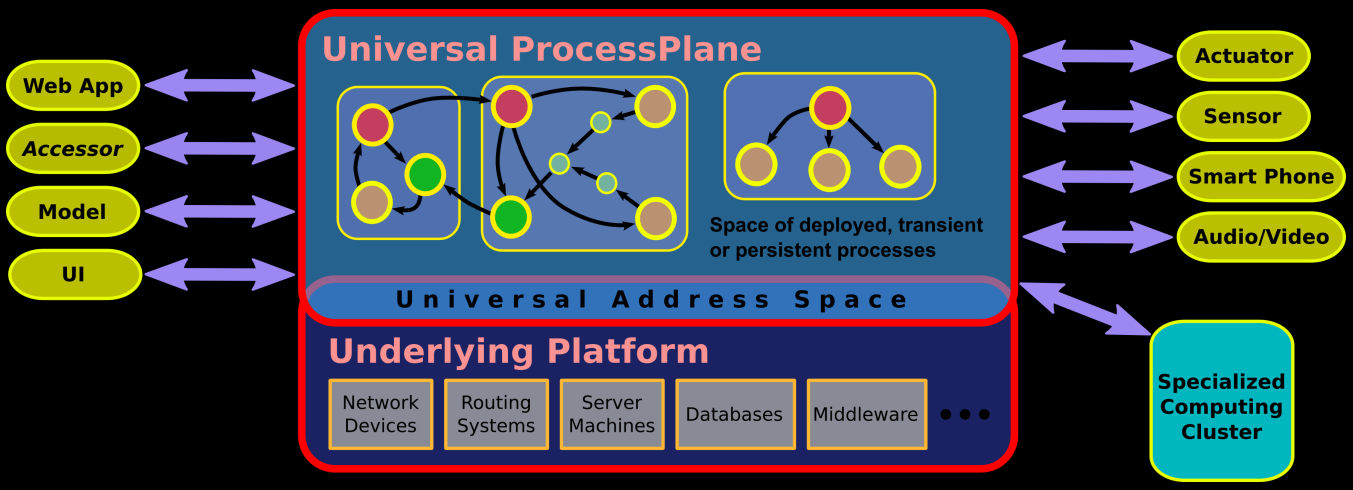
Chris Shaver, Marjan Sirjani

University of California Berkeley, Reykjavik University

The Context

Swarm Applications

Swarm Applications are distributed across a collection of diverse computational devices. Many of these devices are sensors, embedded systems, and mobile entities.

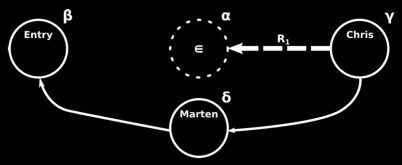


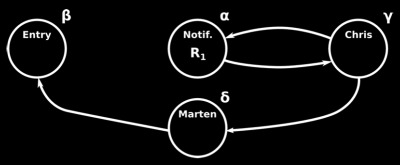
Swarm Applications : Features

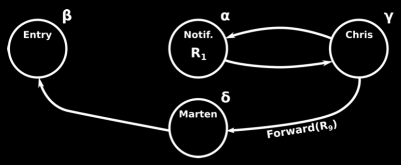
- Concurrent Execution
- Message Sending
- Dynamic Configurability
- Dynamic Process Creation
- Code Mobility
- Semantic Heterogeneity

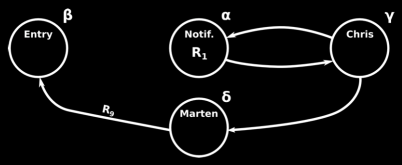
Swarm Processes

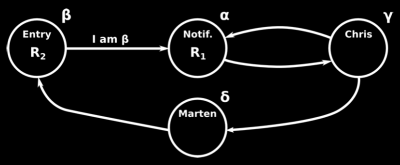
Swarm Processes are collections of behaviors that describe what happens in a Swarm Application.

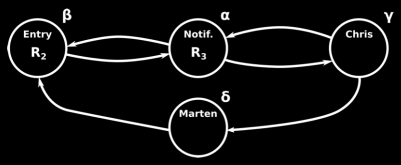












Swarm Processes

Questions :

- How do we describe these processes?
- How do we deal with complex features such as code mobility and topological dynamicity?
- How do we deal with heterogeneity?
- How do we specify, verify, and synthesize?
- What is the logic of these processes?

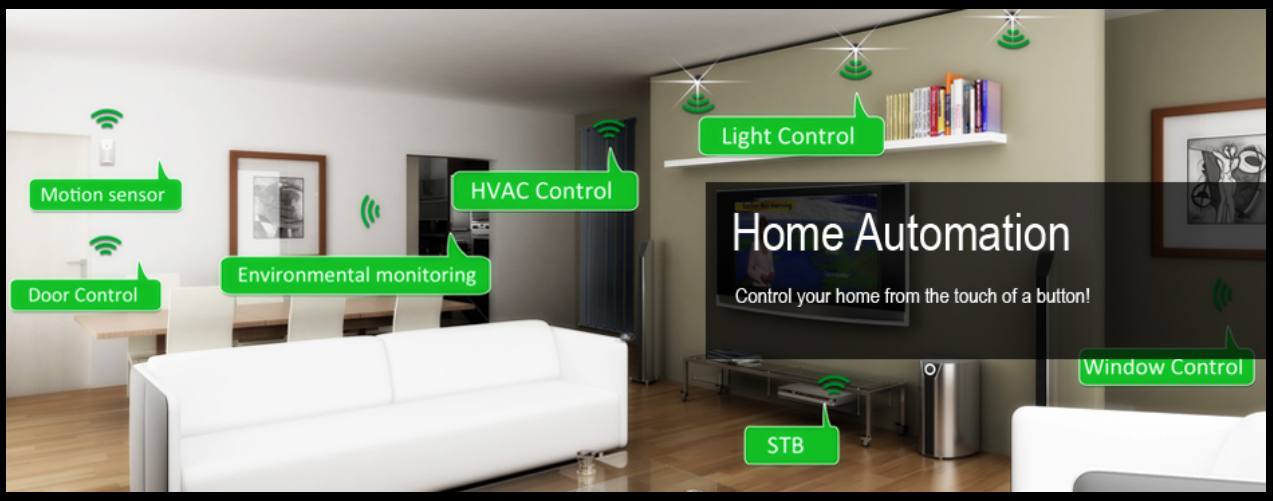
An MoCC for The Swarm

What will this consist of?

- A representation/model for behaviors.
- A formal description of swarm processes in terms of these behaviors.
- A language to describe swarm applications.
- A formal semantics connecting this language to swarm process representations.
- An assertional language that forms observational atomic propositions about swarm processes.
- A logic language (with quantifiers, like LTL) to formulate swarm application requirements/contracts.
- Verification/synthesis tools.

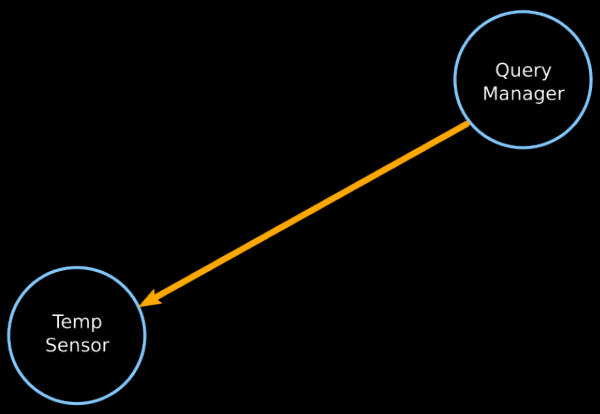
Example: Home Automation

Home Automation



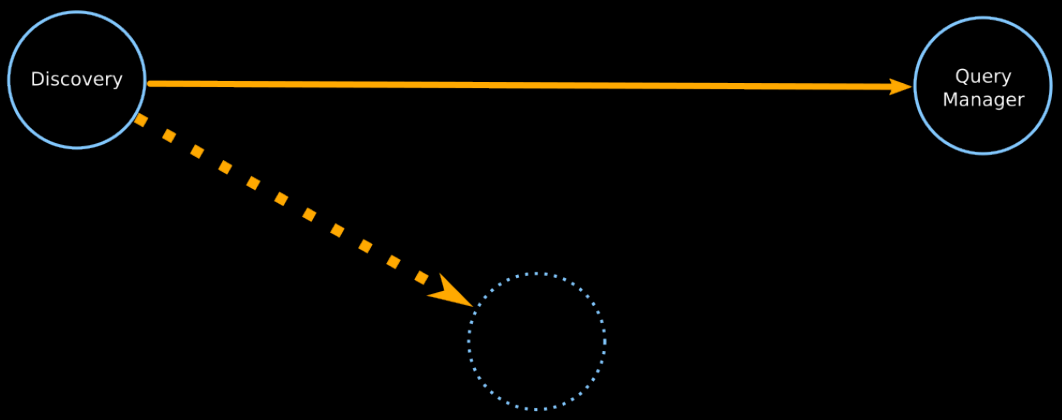
Registering a Device

Registering a Device

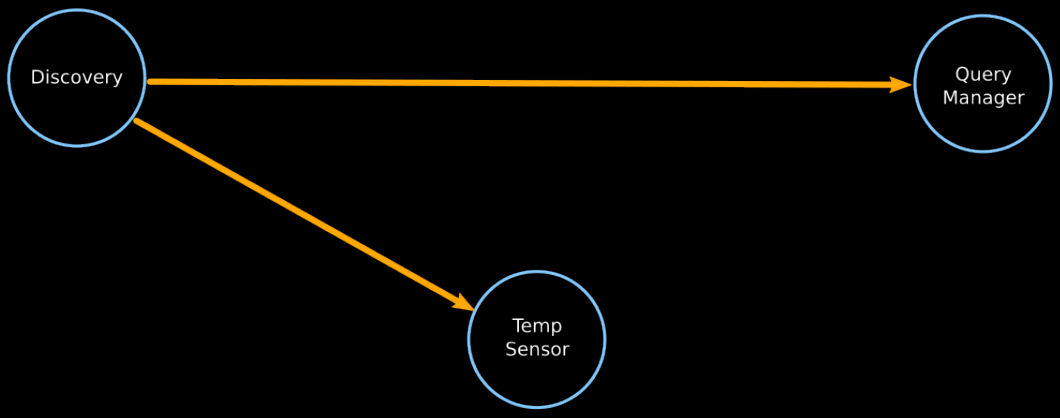


Registering a Device

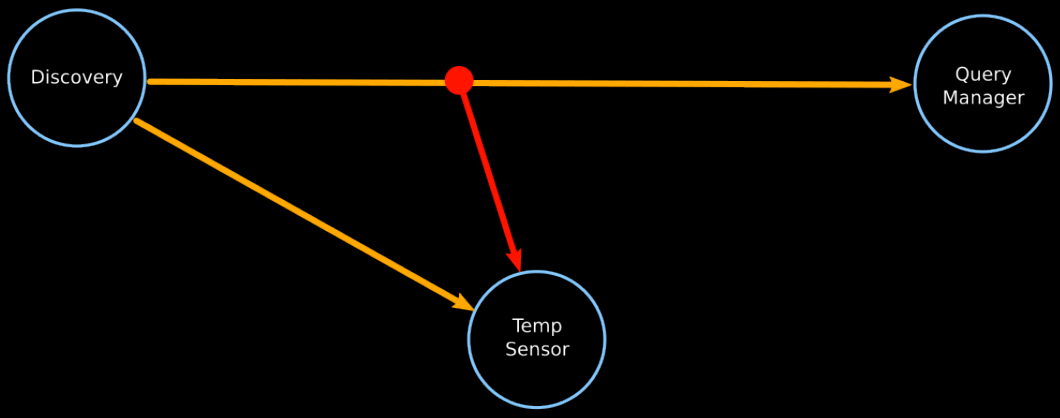
Registering a Device



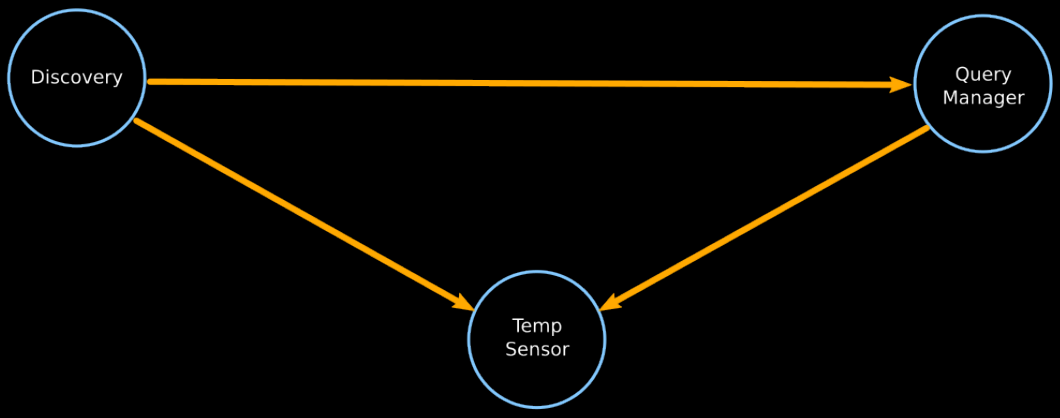
Registering a Device



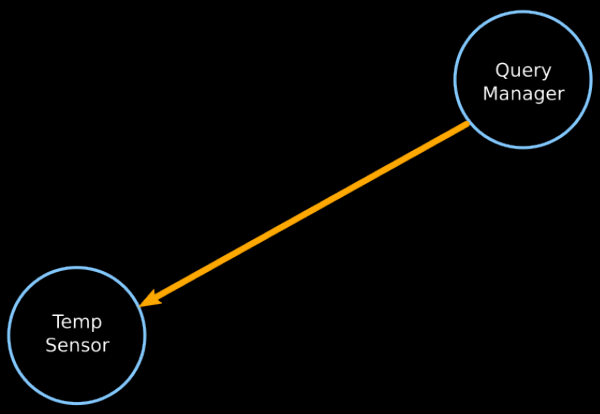
Registering a Device



Registering a Device

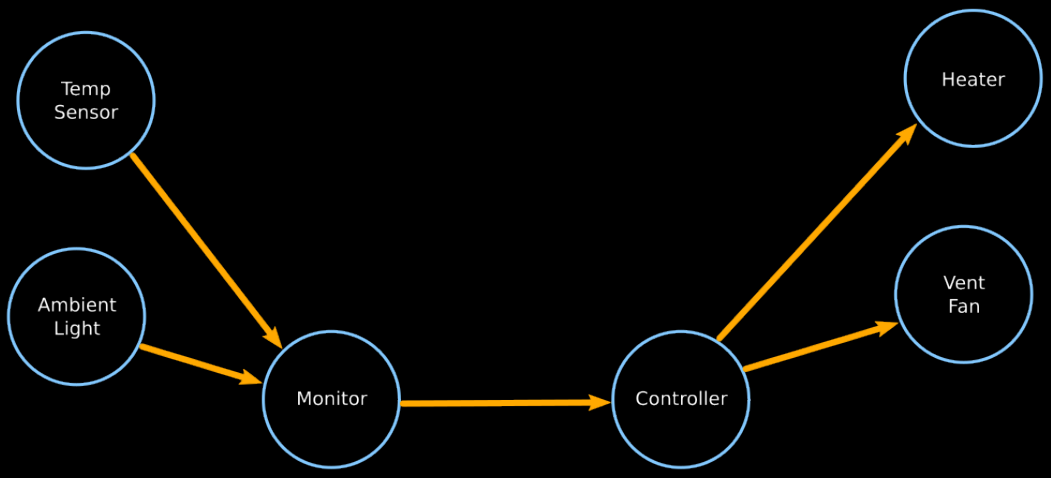


Registering a Device

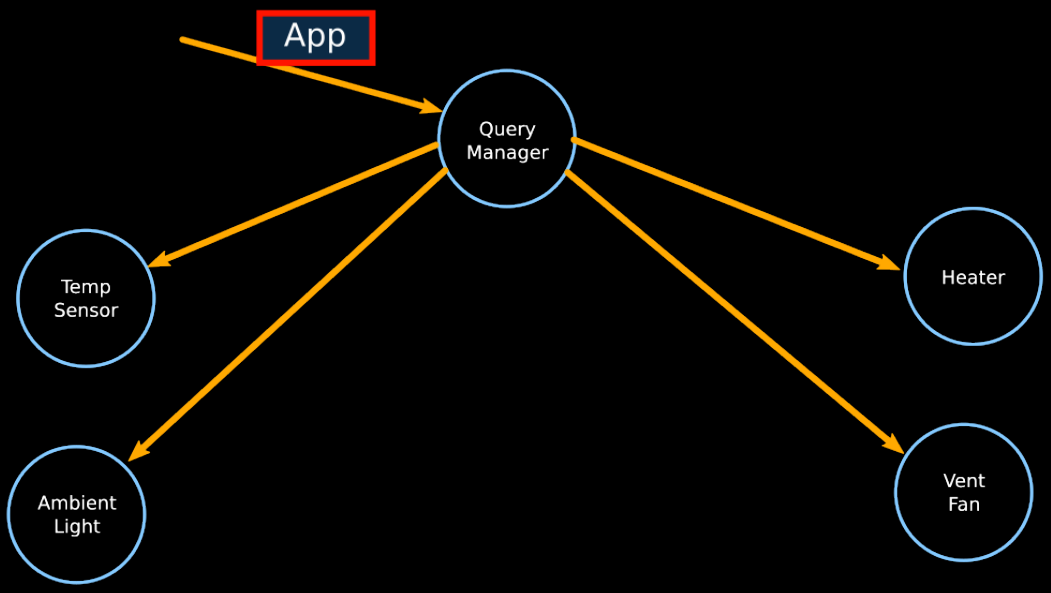


Configuring an Application

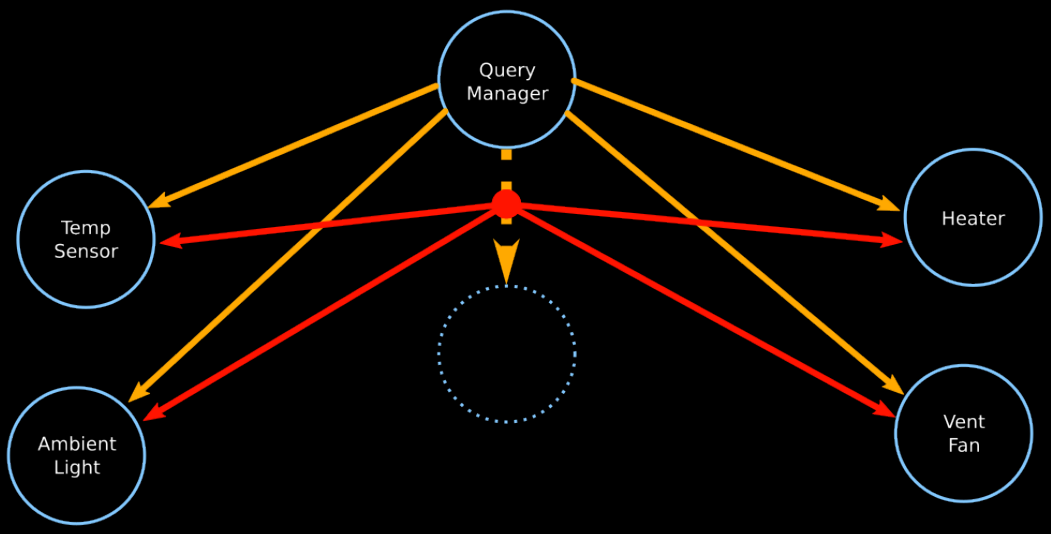
Configuring a Device



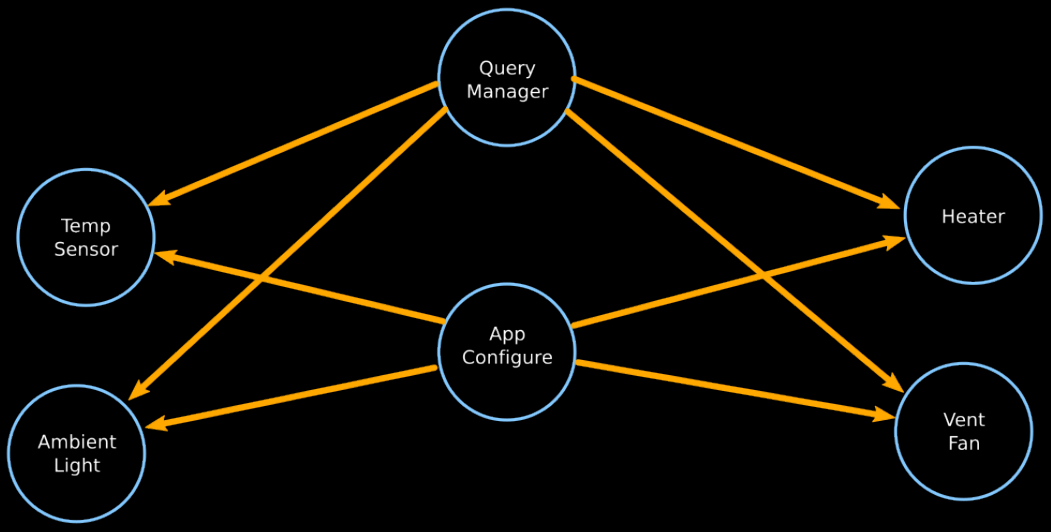
Configuring a Device



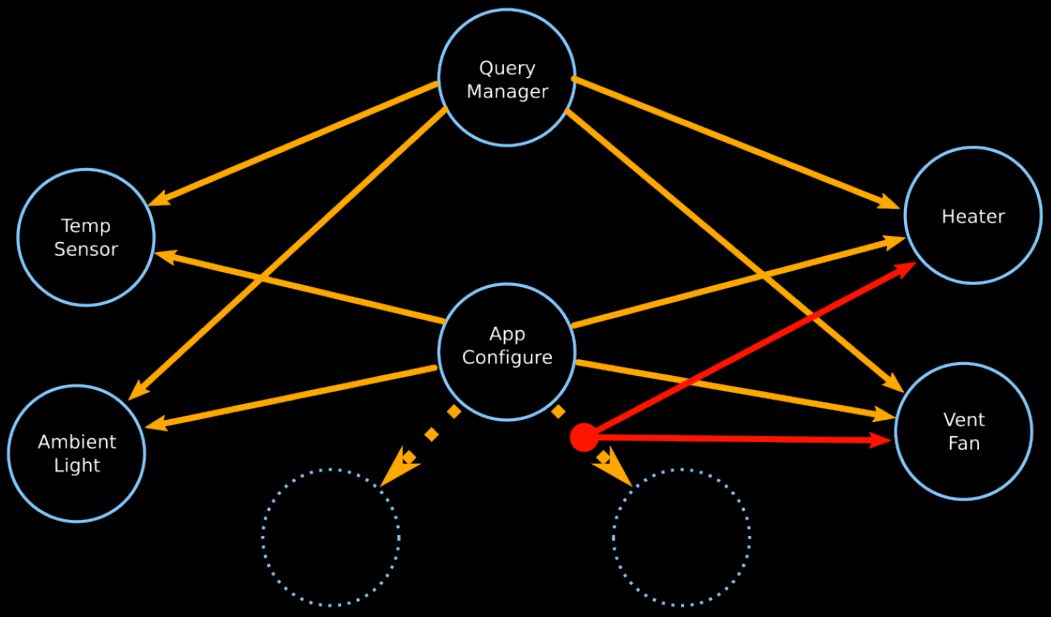
Configuring a Device



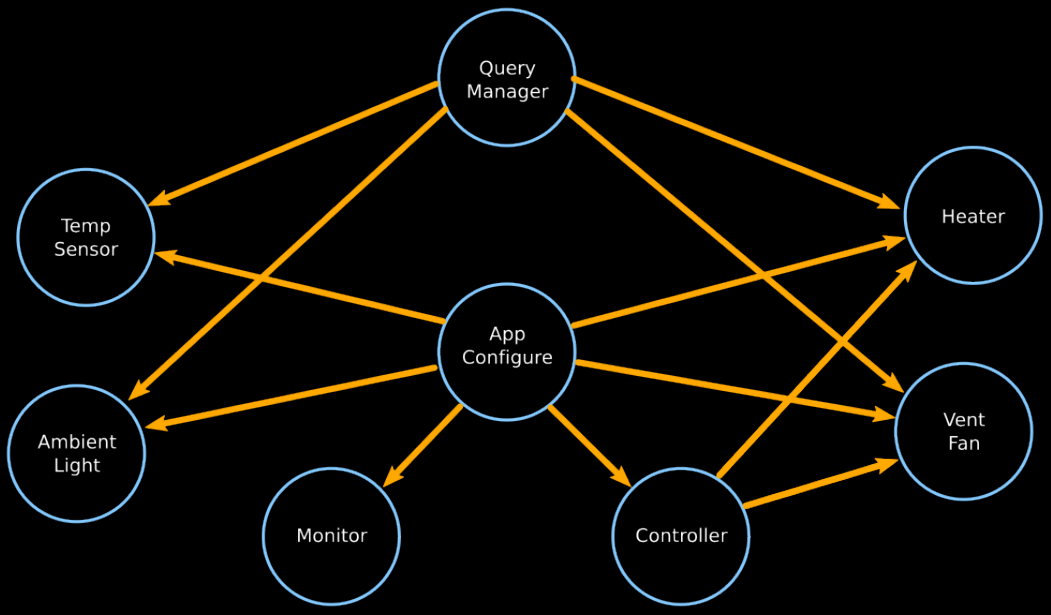
Configuring a Device



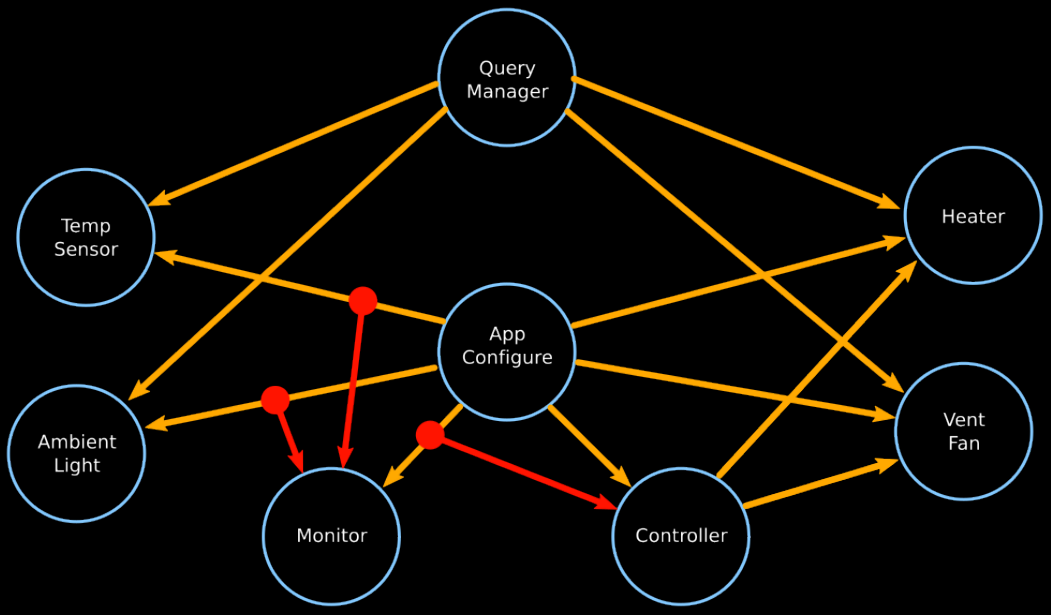
Configuring a Device



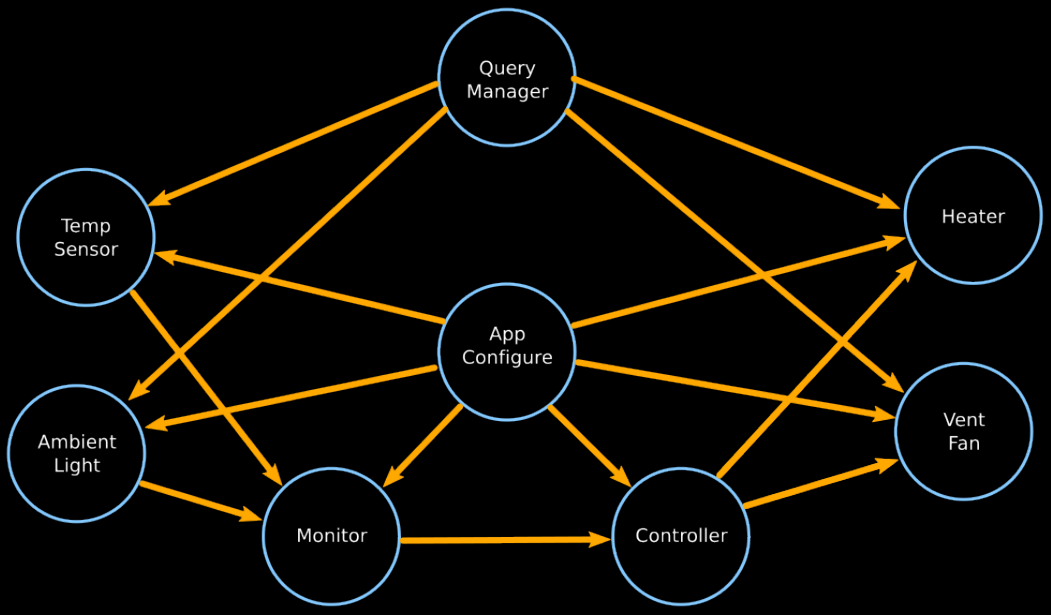
Configuring a Device



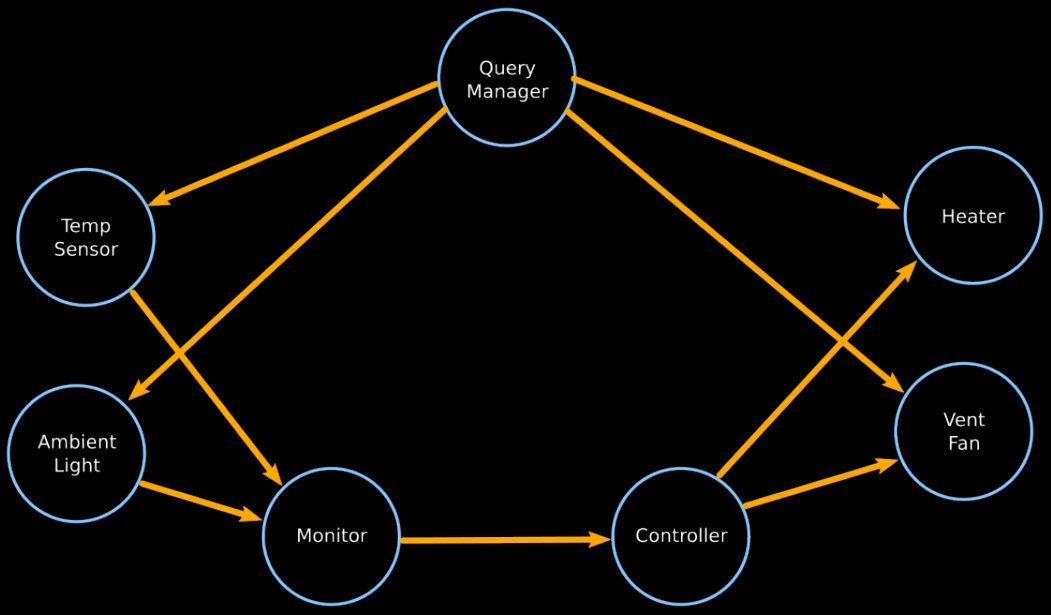
Configuring a Device



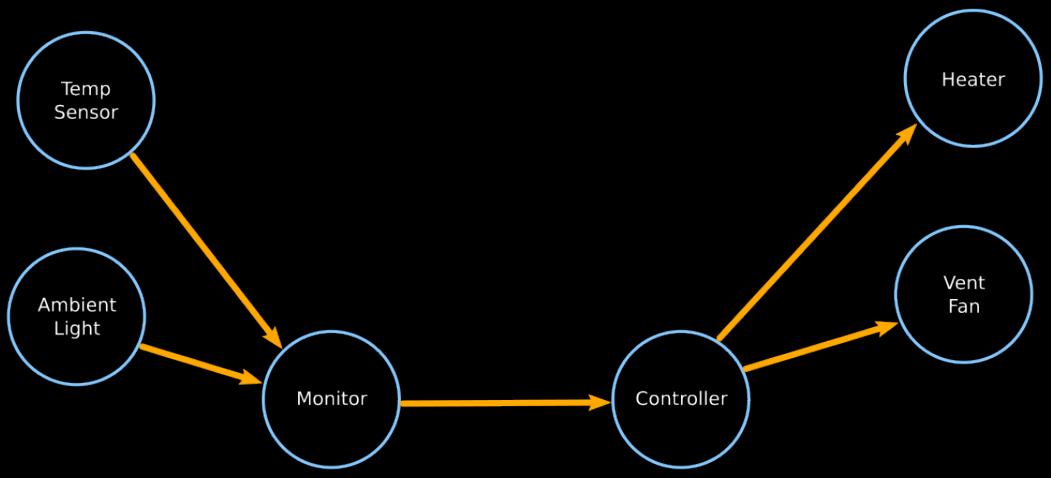
Configuring a Device



Configuring a Device



Configuring a Device



Existing Models

Traces / Interleavings

There is a plethora of literature using traces to develop concurrent semantics. But,

- tailored to a fixed collection of concurrent processes.
- exhaustively overdetermined and processes must be closed over symmetries.
- often assumes a knowledge of states.
- processes often require prefix closures to account for observational states.
- more broadly, observational states are conflated with process states.

And yet, LTL makes this a very supported approach.

Event Systems / True Concurrency

Event Systems are worked on by Winskell et al. and provide a more direct, more ontological model for what happens in a process. It is more attractive that the atomic element is an event, but,

- based on partial orders, hence transitively closed.
- compositions are awkward, because of transitive closure.
- relatedly, dependencies are not directly described in an ontological form.

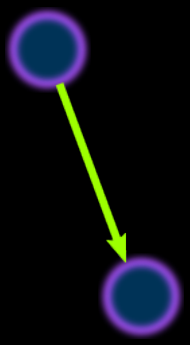
A New Concurrent Representation

Events and Dependencies

Events are unique instances of something happening in a behavior.

- an operation being performed on a machine.
- a message being sent or received
- a reaction to other events: interrupts, application events, etc...
- a physical quantity being measured as having a certain measurement
- a process being created

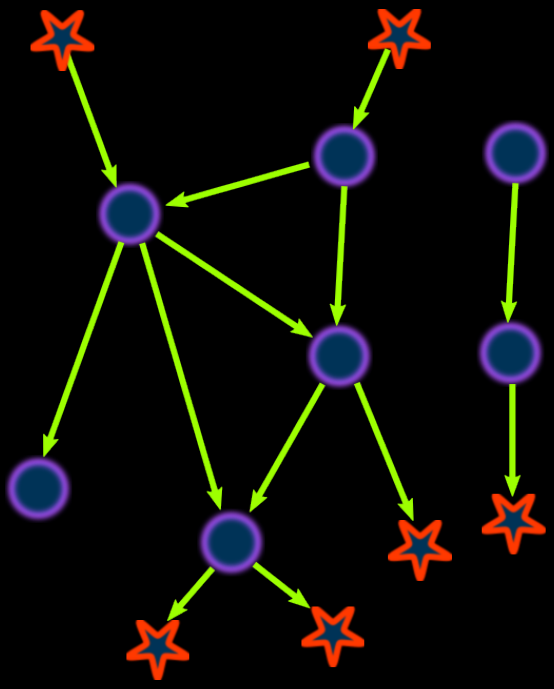
Events and Dependencies



Dependencies indicate conditions or information upon which an event depends as a consequence of another event.

- an operation depending on the conclusion of the previous operation in a sequential process
- a message received depending on a message having been sent
- a sensor reaction depending on a physical quantity passing a certain threshold
- a read-blocked operation depending on the reception of a message
- an event-handler firing depending on the appearance of an event
- a read from a message queue depending on the previous read from the same queue

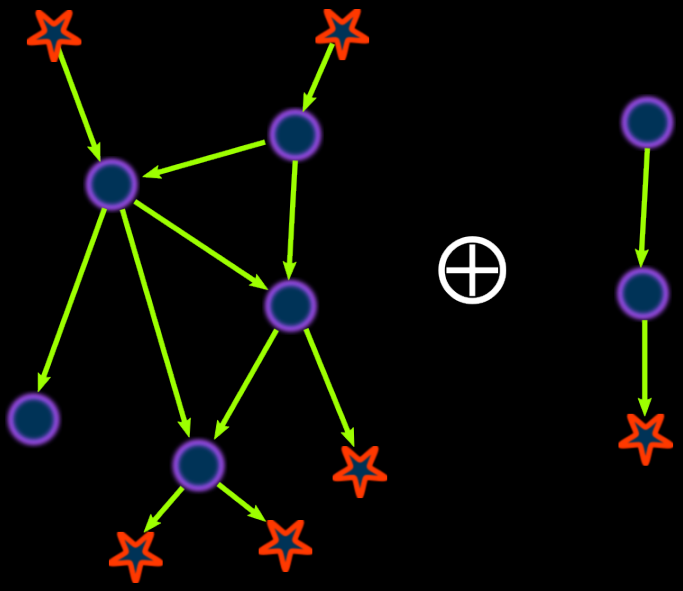
Ontological Event Graphs

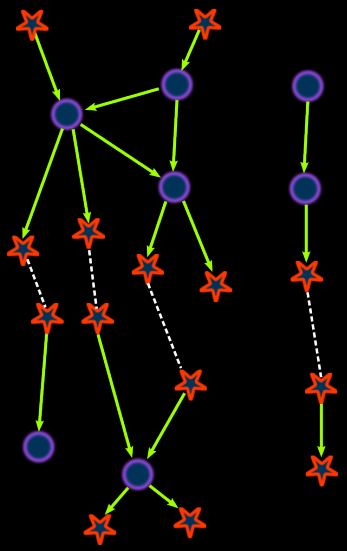


Ontological Event Graphs describe fragments what happened in a Swarm Process, ontologically, independent of the perspective. They consist of events and dependencies. Dependencies can either be connected to an event or open ended (indicated by the star), suggesting and incoming or outgoing connection to preceding or following events.

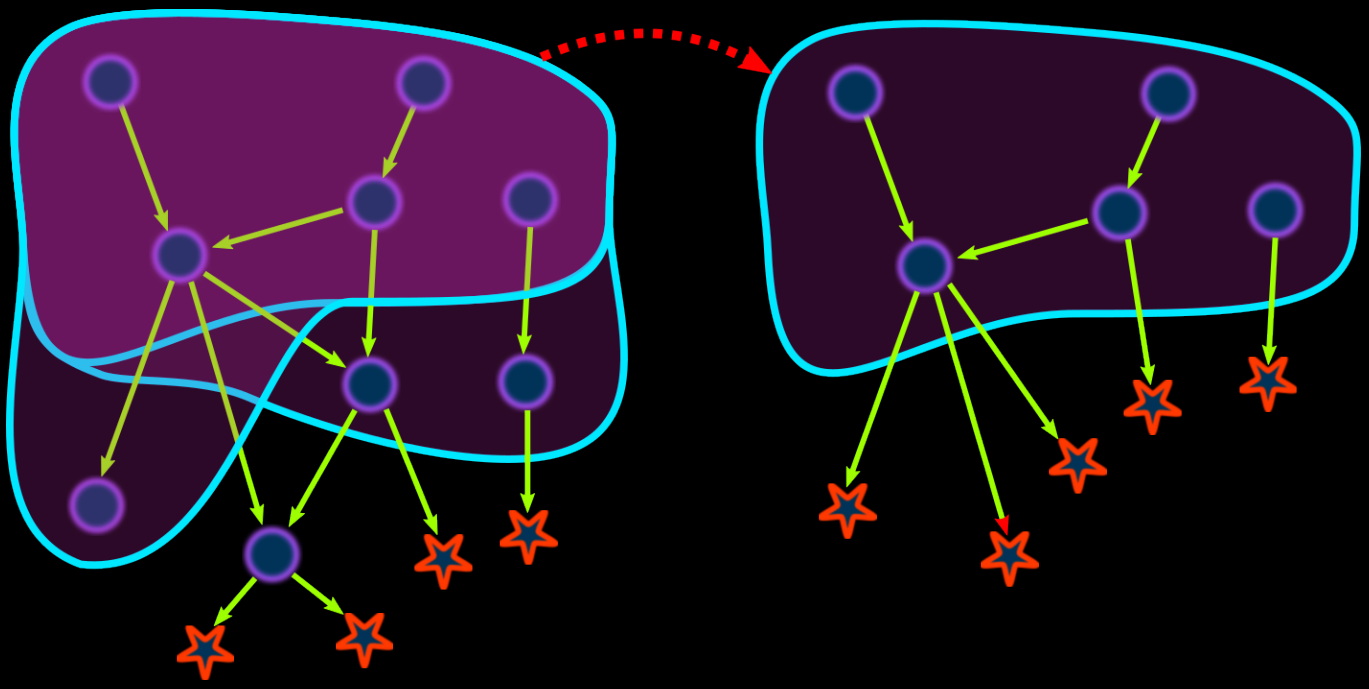
OEG Composition

OEGs can be composed in parallel or sequentially by attaching open dependencies. This can be typed, of course.



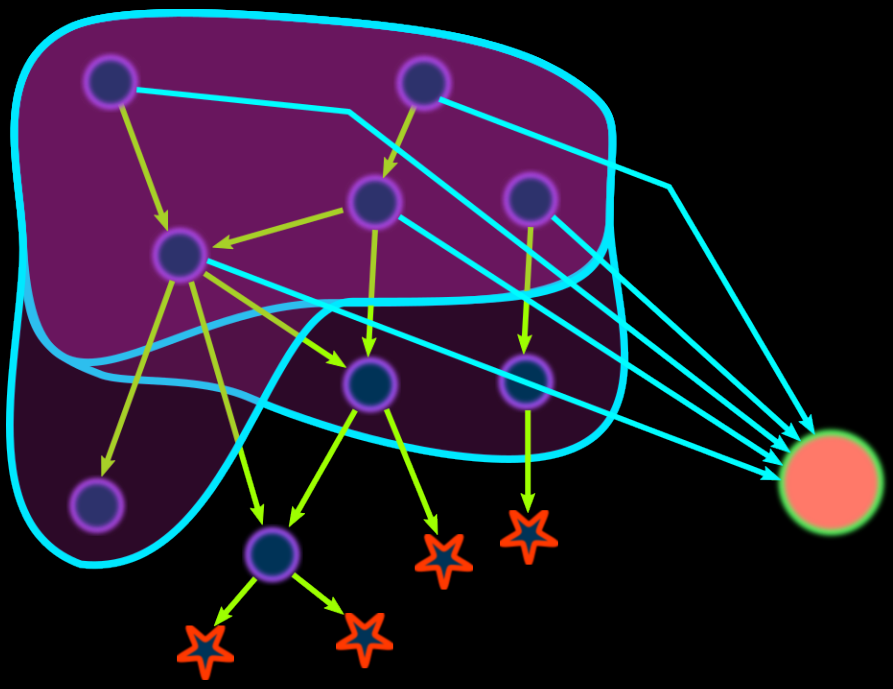


OEG Prefixes

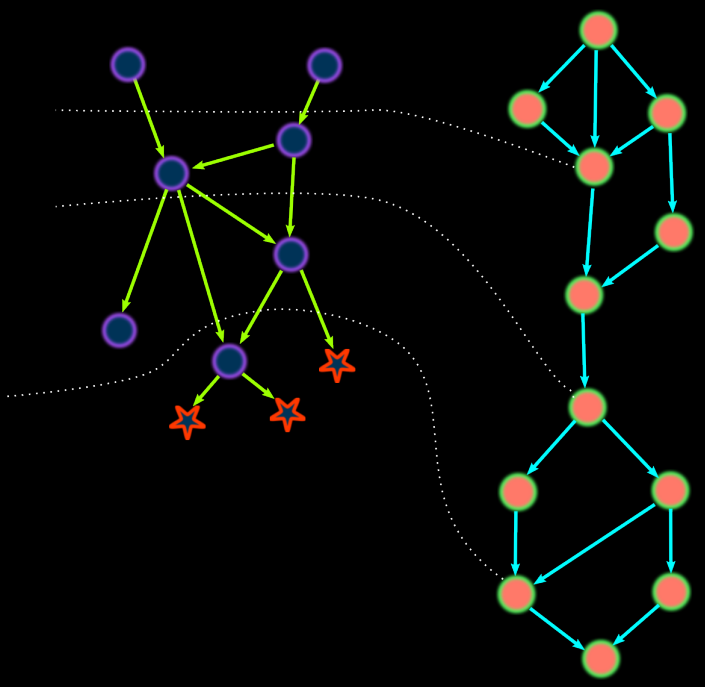


Perspective

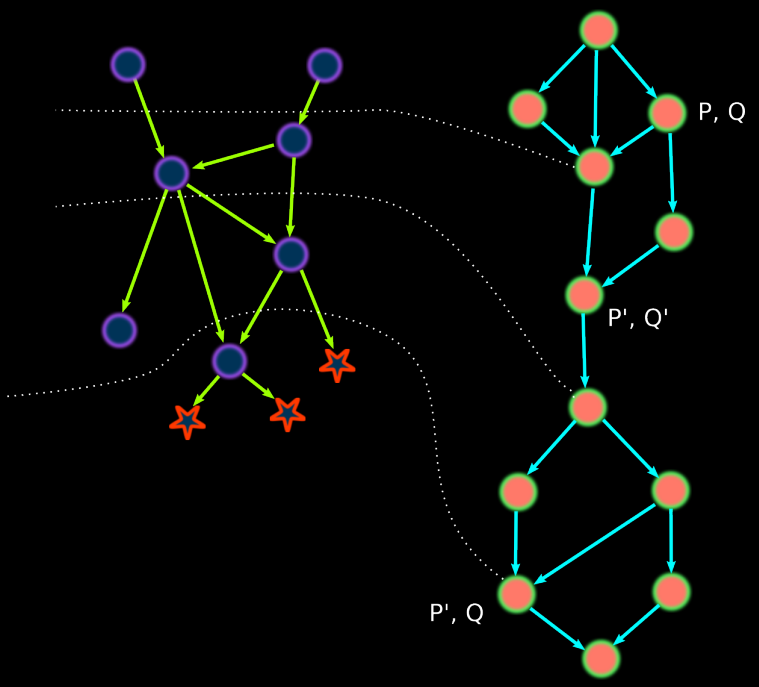
Observations



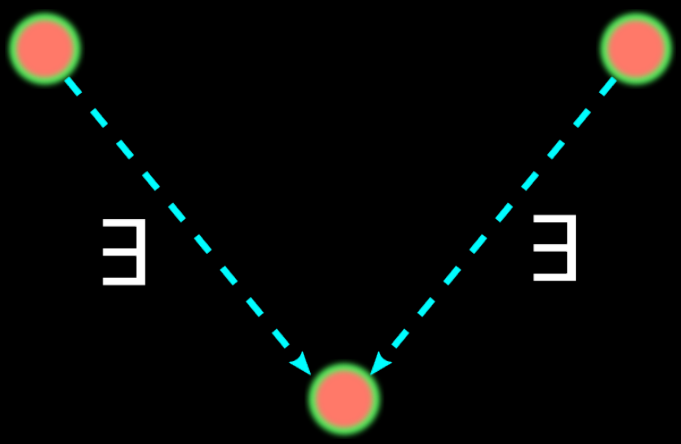
Epistemological Event Graphs



Epistemological Event Graphs



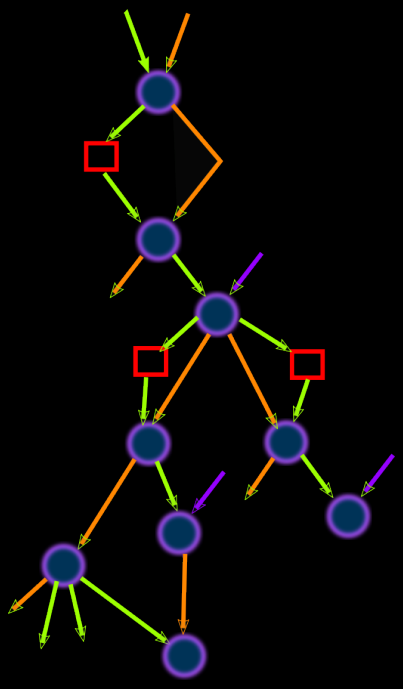
Confluence



For any two observational states, there are always exist paths to bring them into a common confluent observational state. This is like the Church-Rosser property!

Example: Home Automation

Configuring an Application



Here, an OEG can be given for a fragment of behavior associated with the actions in the configuration example. Orange arrows are control dependencies, green arrows are communication, and purple arrows are initialization. Red boxes are allocation events.

Conclusions

Progress on this work.

- Currently working on the mathematical details.
- Planning software tools to generate and manipulate OEGs and EEGs
- More examples!?
- Develop languages of assertion and logic formulas.
- Connecting with language design work: ReActors.

Thanksabunch!