Verification of Orbitally Self-Stabilizing Distributed Algorithms using Lyapunov Functions and Poincaré Maps

Abhishek Dhama, Jens Oehlerking, Oliver Theel

April 3, 2006

Outline

- Distributed Algorithms
- Self-Stabilization
- Example Algorithm
- Parallels to Stability of Control Systems
- Application to Example Algorithm
- Conclusion/Future Work

Distributed Algorithms

A number of processes of the form

process P_i

var $x_1, x_2, x_3, \ldots$: integer

begin

$$\begin{array}{lll} g_1(x_1, x_2, x_3, \ldots) & \longrightarrow & C_1 \\ [\!] \quad g_2(x_1, x_2, x_3, \ldots) & \longrightarrow & C_2 \\ [\!] \quad g_3(x_1, x_2, x_3, \ldots) & \longrightarrow & C_3 \\ \vdots & & \end{array}$$

end

which communicate with each other, by using dedicated communication registers

Execution Semantics

A so-called *daemon* picks a guard that evaluates to true (i.e. the guarded command is *active*) in a *fair* manner. Its *assignment statement* is then executed atomically

Possible execution semantics:

- maximum parallel semantics: pick one active guarded command per mode and execute simultaneously
- serial semantics: pick one active guarded command from the entire system and execute it, no parallel execution possible
- and others....

here: maximum parallel semantics

Self-Stabilization

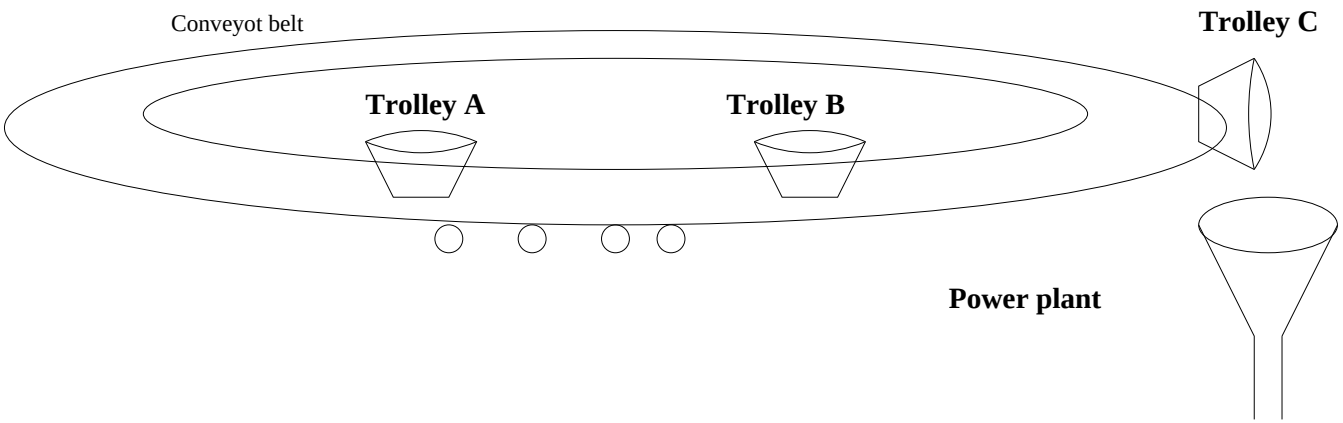
A distributed algorithm is called *self-stabilizing* wrt. to a predicate P if from any starting state:

- a state fulfilling P is reached in finite time
- and not left again voluntarily

Starting state is used to model transient faults. If P represents "legal" state of the system, then self-stabilization wrt. P implies that the system can recover from any such fault.

Example Algorithm

Circular conveyor-trolley system in power plant



Robot (not pictured) synchronizes trolleys with powerplant operations through internal counters.
If it detects a trolley out of synch, counters are updated by the robot.
$\Rightarrow$ eventually synchrony will be achieved

Example algorithm (continued)

Algorithm is modelled with

- one process per trolley
- one process for the robot

Robot process communicates with one trolley process at a time, attempting to synchronize with it.

System state is modelled through variables

- $X_T[i]$ for each trolley process, representing its position
- $X_R[i]$, one per trolley, for the robot process, representing the robot's view of the trolley position
- $LX_{RT}[i]$ and $LX_{TR}[i]$ representing the communication links

Goal: synchronize the $X_T[i]$ with the $X_R[i]$.

Modeling Distributed Algorithms as Hybrid Systems

Model as hybrid system with nondeterministic switching:

$$\begin{aligned} x[k+1] &= f_m(x[k]), m \in \mathcal{M} \\ \Phi &\subseteq \mathbb{R}^n \times \mathcal{M} \times \mathcal{M} \end{aligned}$$

One mode represents one guarded command, where

- $f_m(x)$ represents the assignment statement
- Φ is chosen such that the activation of guards is modeled, i.e., $(x, m_i, m_j) \in \Phi$ iff $f_{m_i}(x)$ fulfills the guard of the guarded command corresponding to m_j

Modeling the Example Algorithm

Mode dynamics look like this:

$$\begin{bmatrix} X_R[0]^+ \\ LX_{RT}[0]^+ \\ X_T[0]^+ \\ X_R[1]^+ \\ LX_{RT}[1]^+ \\ X_T[1]^+ \\ \vdots \\ X_R[n-1]^+ \\ LX_{RT}[n-1]^+ \\ X_T[n-1]^+ \\ next^+ \end{bmatrix} = \begin{bmatrix} (X_R[0]+1)\%n' \\ LX_{RT}[0] \\ (X_T[0]+1)\%n' \\ (X_R[1]+1)\%n' \\ LX_{RT}[1] \\ (X_T[1]+1)\%n' \\ \vdots \\ (X_R[n-1]+1)\%n' \\ LX_{RT}[n-1] \\ (LX_{RT}[n-1]+1)\%n' \\ (next+1)\%n' \end{bmatrix}$$

Orbitally Self-Stabilizing Algorithms

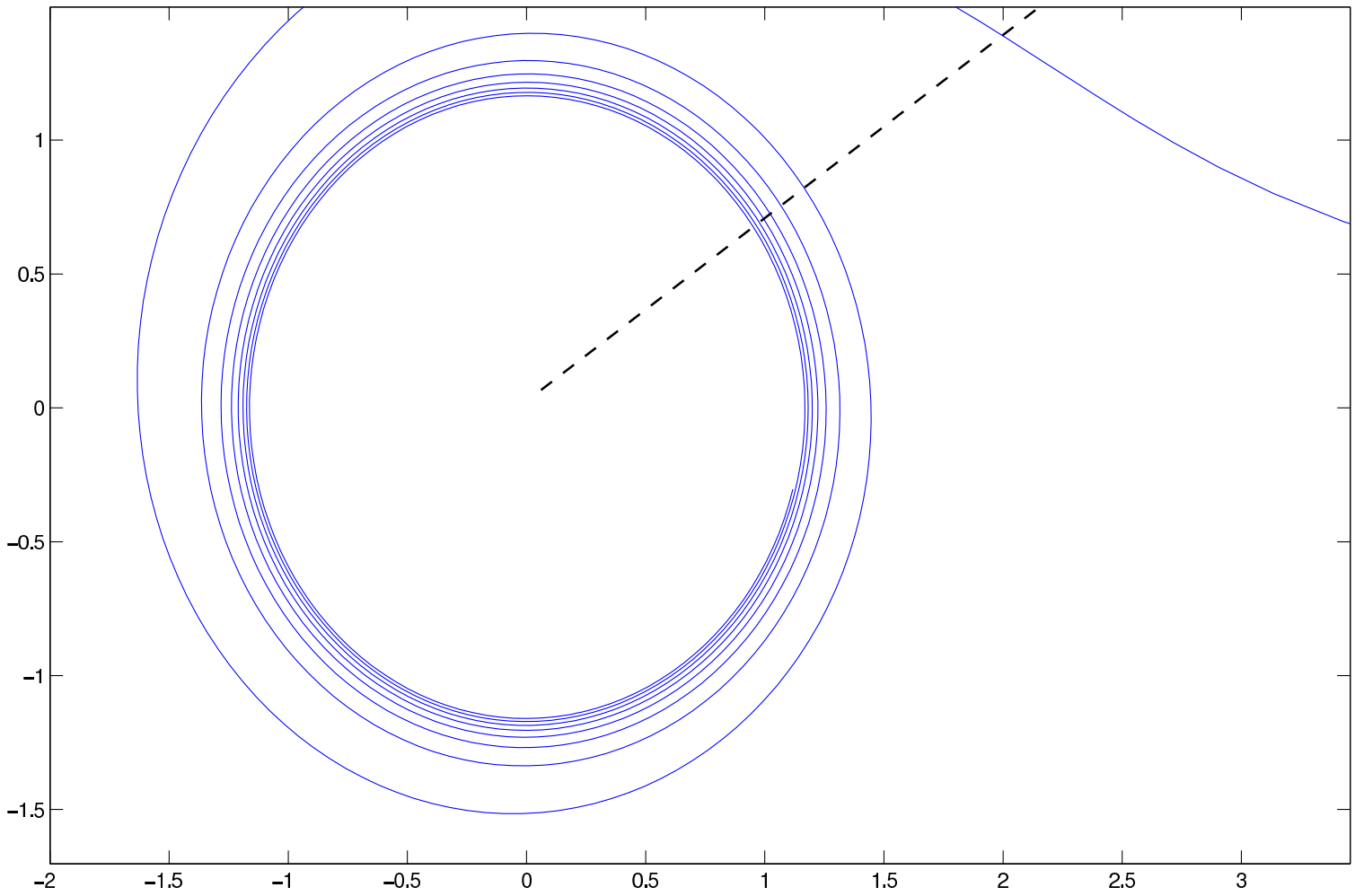
Set of legal states forms a cycle.
Want to prove: Algorithm converges toward cyclic behavior.

orbital self-stabilization of a distributed algorithm

$\Updownarrow$

stability wrt. a limit cycle of a hybrid system

Control theoretic method: *Poincaré map*.


1
0.5
0
−0.5
−1
−1.5
−2
−1.5
−1
−0.5
0
0.5
1
1.5
2
2.5
3

Poincaré Map for Example Algorithm

Communication pattern is periodic $\Rightarrow$ Poincaré map can be calculated by n-times composition of hybrid system dynamics.
For trolley i:

$$\begin{bmatrix} X_R[i]^+ \\ X_T[i]^+ \end{bmatrix} \in \left\{ \begin{bmatrix} X_R[i] \\ X_T[i] \end{bmatrix}, \begin{bmatrix} (X_R[i]-2)\%n' \\ X_T[i] \end{bmatrix}, \begin{bmatrix} X_R[i] \\ (X_T[i]-2)\%n' \end{bmatrix} \right\}$$

Last step: Show stability of difference inclusion
$\Rightarrow$ Lyapunov functions

Stability of Poincaré Map

Find a Lyapunov-like function for the differential inclusion, i.e. a function $V : \mathbb{R}^n \to \mathbb{R}$ with:

- $V(x) > 0$ for all x not on the limit cycle
- $V(x) = 0$ for all x on the limit cycle
- $V(x[k+1]) \leq V(x[k])$ for all $x[k]$ not on the limit cycle

For the example algorithm:

$$V_i(X_R[i], X_T[i]) = X_R[i] + X_T[i] + n' * (X_R[i]\%2 + X_T[i]\%2)$$

Conclusion

We have shown:

- how distributed algorithms can be modelled as hybrid systems
- that there is a parallel between self-stabilization of such algorithms and stability of control systems
- how Poincaré maps and Lyapunov functions can be employed to prove self-stabilization

Current/Future Work

Apply method to algorithms with:

- different execution semantics
- no dedicated "server" process
- more general communication structure (general graph)
- different kind of self-stabilization (to a single state, to a rectangular set of states)

Questions?