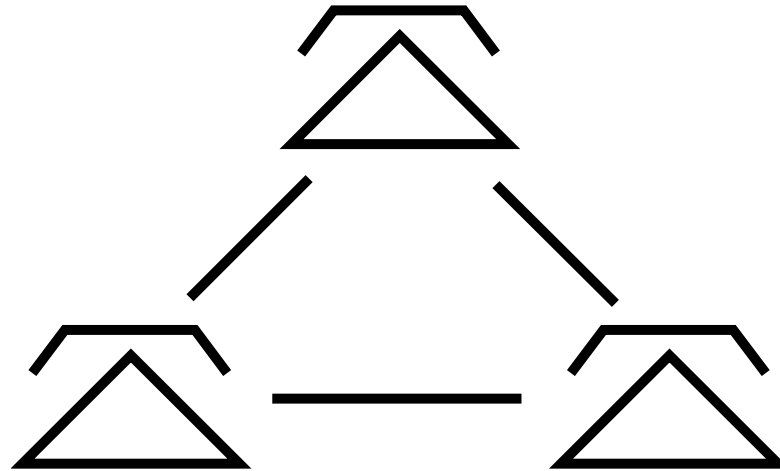


Blocking in Networks of Arbitrary Size

Siamak Nazari & John Thistle
Thales Group, Toronto
& Dept. of ECE
University of Waterloo
Canada

`jthistle@kingcong.uwaterloo.ca`

Motivation



- ‘feature interaction’ in telecommunications networks

Problem

In a network consisting of an arbitrary number of identical, or at least isomorphic, finite-state DES, check whether:

- component DES are potentially blocked from reaching their marked states (“component blocking”);
- or whether the network as a whole is potentially unable to reach a state in which all components are in marked states (“network blocking”).

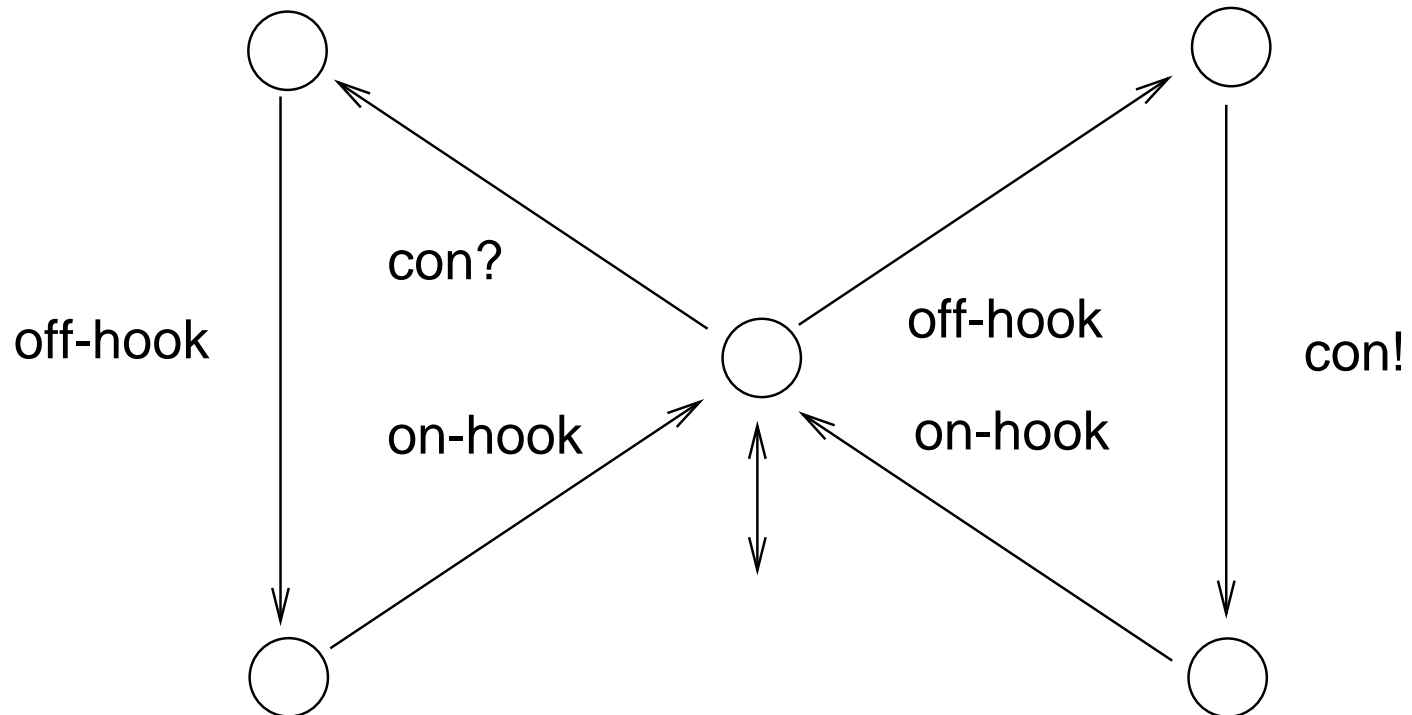
Background:

- instance of 'parameterized model-checking problem'
- generally undecidable (MTNS '04)
- semi-decision procedures based on process equivalences (CDC/ECC '05)

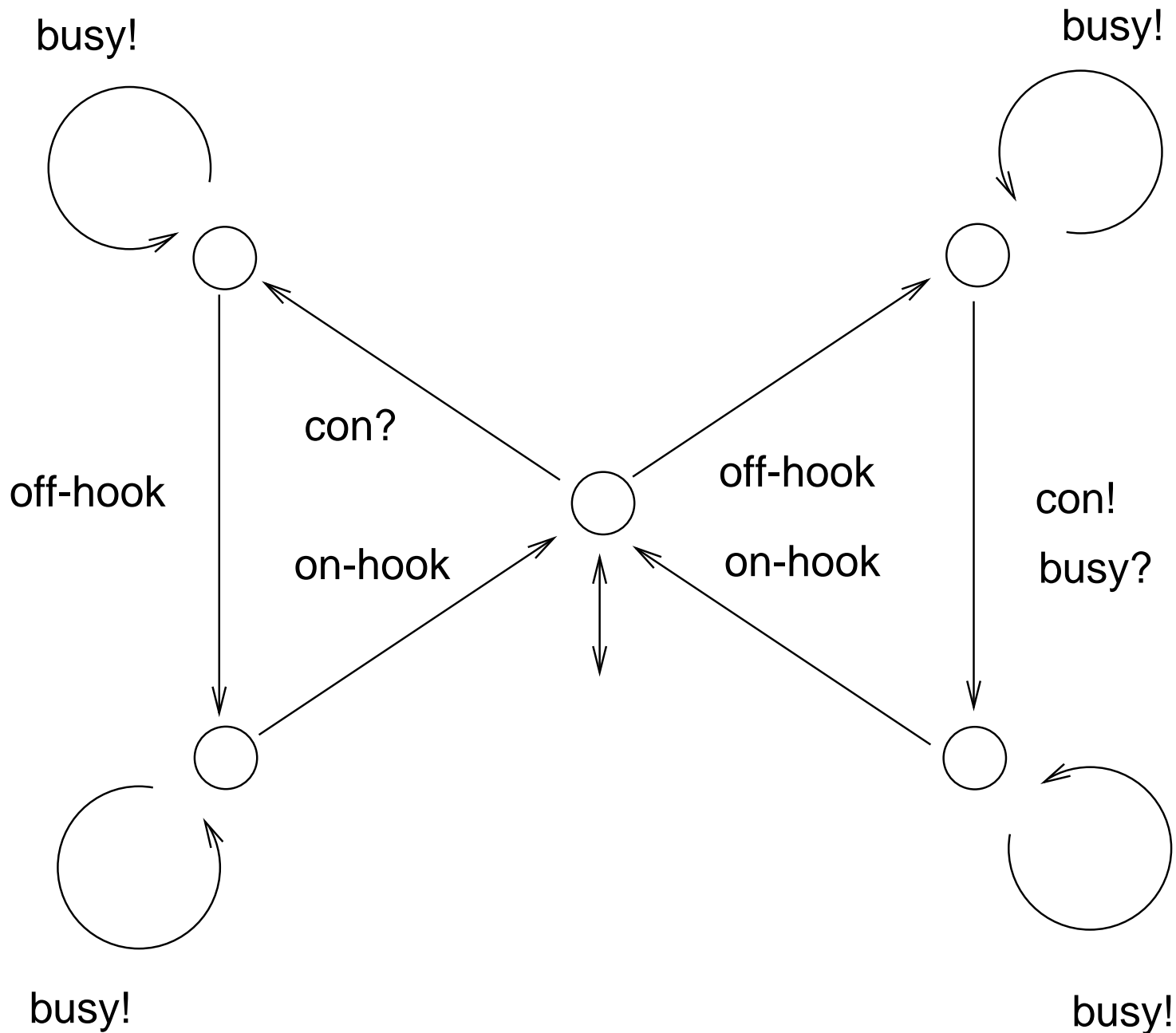
This paper:

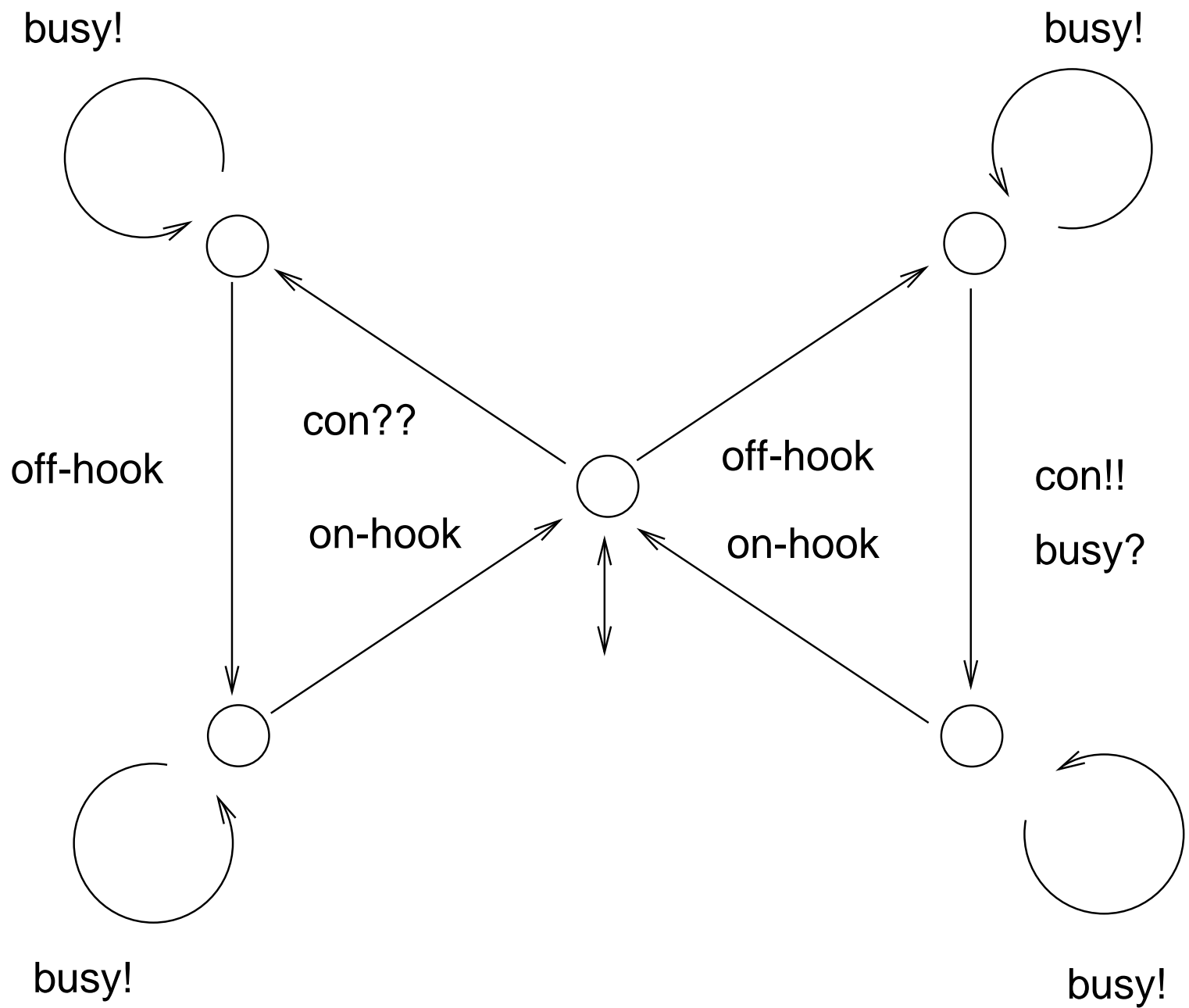
- undecidable for identical finite-state processes in presence of both “rendezvous” and “broadcast” actions;
 - decidable for identical finite-state processes interacting only via “rendezvous”;
 - undecidable (even w/o broadcast) for isomorphic finite-state processes . . .
- . . . but if a given process interacts, at a given time, with only a bounded number of others – decidable.

Simple Example:



Identical processes, with rendezvous actions.
Processes block if all attempt to place calls.



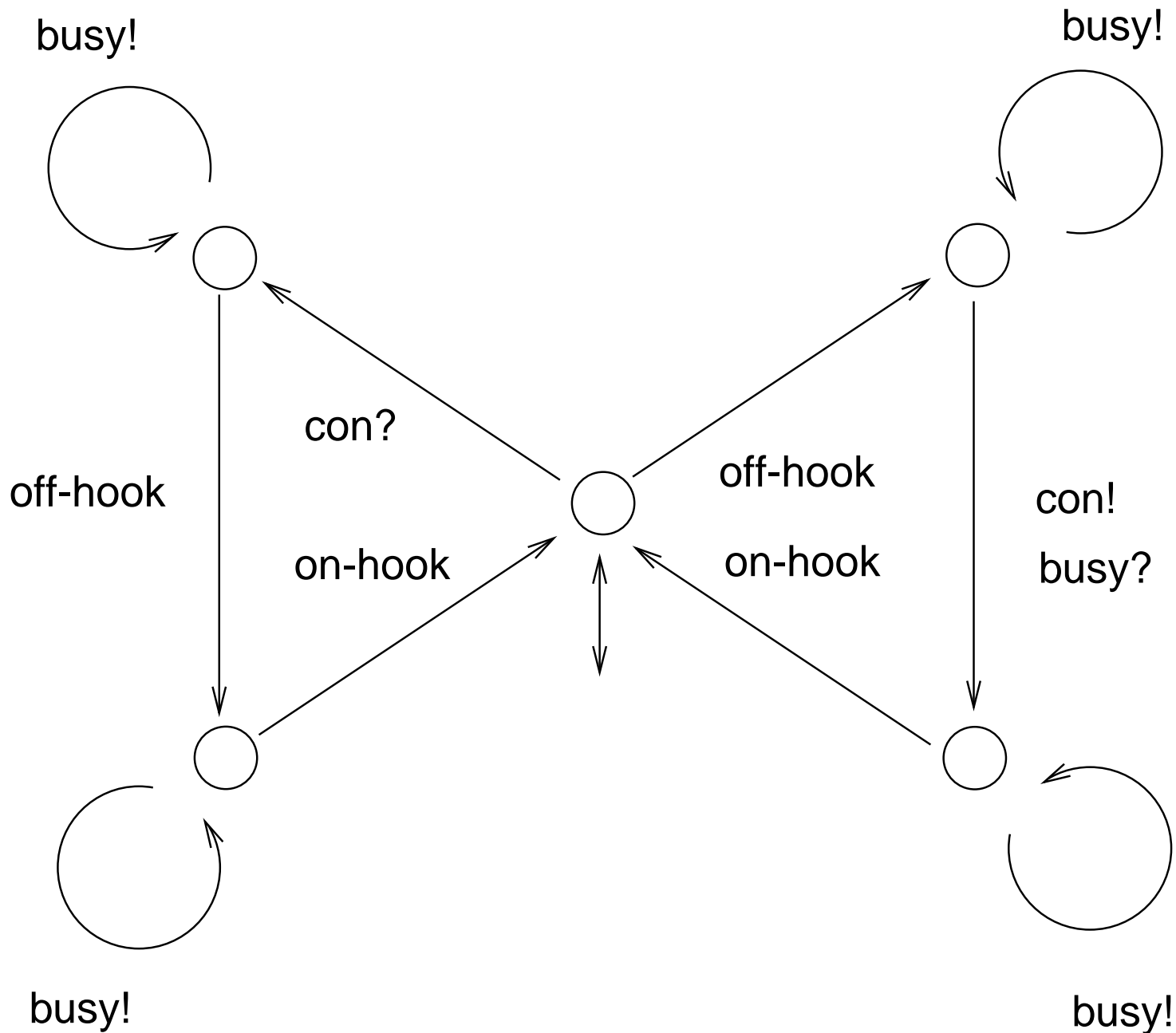


Checking for blocking is **decidable** for identical finite-state processes **in absence of broadcast**:

- by reduction to home-space problem for Petri nets
- if putative home space is “semilinear,” this problem is decidable.

Let's consider **network blocking**.

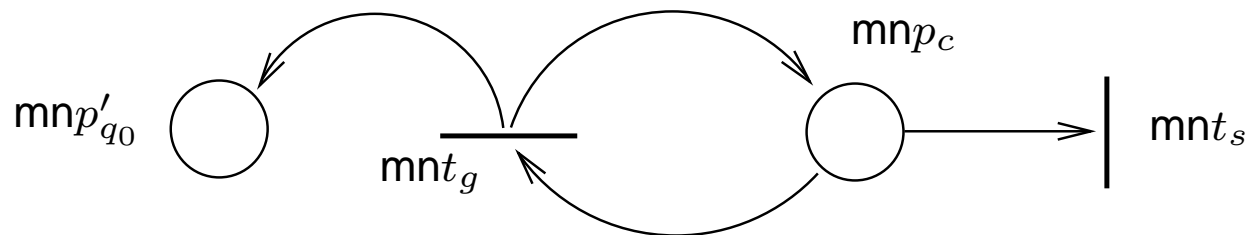
The idea is to model the network as a Petri net . . .



... (states become places and actions become transitions);

... rendezvous is captured by “merging transitions” in the usual Petri-net manner;

... add mechanism to inject an arbitrary number of tokens into the “initial state” place ...



... 'home space' is set of markings where places representing unmarked states have no tokens ...

... this set of markings can easily be seen to be semilinear.

Proof for **component blocking** is essentially similar.

Note: **with broadcast actions**, this construction doesn't work . . .

. . . we need inhibitory arcs to model broadcast, and that rules out decidability.

In fact, in the presence of broadcast actions, checking blocking becomes **undecidable**.

Proof idea: reduction from the halting problem for **2-counter machines**.

A **2-counter machine** consists of nonnegative-integer-valued counters c_1, c_2 and a program with a finite number r of instructions.

Each instruction takes one of the following forms:

$inc(c_i)$: **increments** counter c_i and advances to next instruction;

$dec(c_i)$: **decrements** c_i and advances to next instruction;

$jump(c_i, j, k)$: if $c_i = 0$, **jump** to j^{th} instruction, else to k^{th} ;

$halt$: the program **halts**.

The eventual halting of such a machine with its counters initialized to zero is **undecidable**.

Given such a machine, we can construct a process such that deciding nonblocking for a network of arbitrary size would allow decision of halting of machine.

States of process: $I, D, E, H, c_1, c_2, s_0, s_1, \dots, s_{r-1}$

Initially, all processes in state I ;

Via a broadcast action, one process evolves to state s_0 (becoming “control process”) . . .

. . . all others evolve to state D (become “drones”);

idea is that counter values will be represented by numbers of drones in states c_1, c_2 .

E and H will represent “error” and “halting” states.

Counter **incrementing** is simulated by a local action that moves a drone from state D to c_i ;

decrementing by a one that moves a drone from state c_i to D .

Jumps use a broadcast event: this moves the control process to state s_j ; if any drones are in state c_i , this same event moves them to the error state E , where they remain; . . .

. . . alternatively, the control process can execute a local action that takes it to state s_k , but must be synchronized with a drone process in state c_i .

A **halt** moves the control process to the halt state H , which is the only unmarked state.

It follows that the 2-counter machine halts only if a (sufficiently large) network exhibits component blocking

...

... and conversely, a network exhibits component blocking only if the machine halts.

In some instances, modelling requires that it be possible for one process to interact with a specific other process – e.g. by indexing processes.

In such cases, processes are **isomorphic**, but not identical.

Even in the absence of broadcast, this makes blocking undecidable: proof is by reduction from the halting problem for Turing machines (as in MTNS '04).

CDC/ECC '05 paper gave semidecision procedures.

Current paper exploits special structure:

- in motivating examples, processes interact only with a bounded number of others, then 'reset,' erasing all memory of past interactions.
- a 'template protocol' ensures decidability of blocking in such cases.

Conclusions:

We've looked at one class of distributed (?) DES problems.

We've focused on decidability: not necessarily a make-or-break issue, but one indication of mathematical structure, or lack thereof.

In the end, we seem to have arrived at a means of ensuring decidability for a sufficiently rich class of models.

In the long run, some other formulation, such as formal logic, may prove more helpful for addressing "parameterized" systems.

Working on merging decidability results with work on synthesis to synthesize nonblocking control.