Anomaly detection in event-based manufacturing systems using model generation

Dawn Tilbury

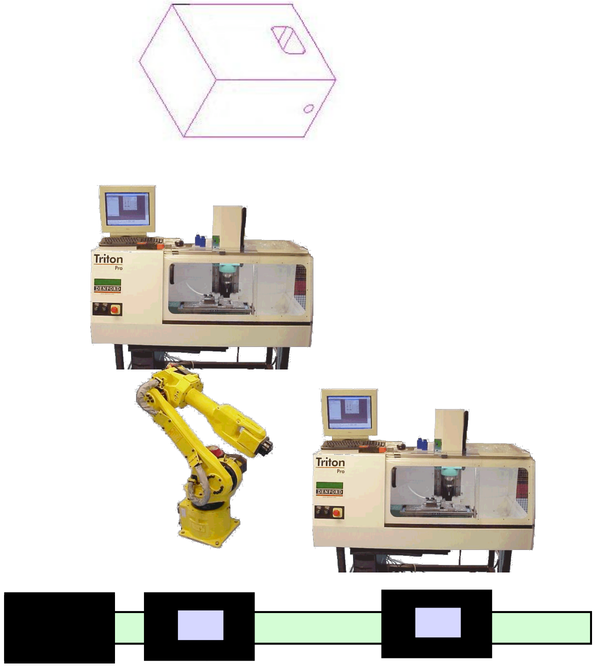
Professor, Mechanical Engineering, University of Michigan

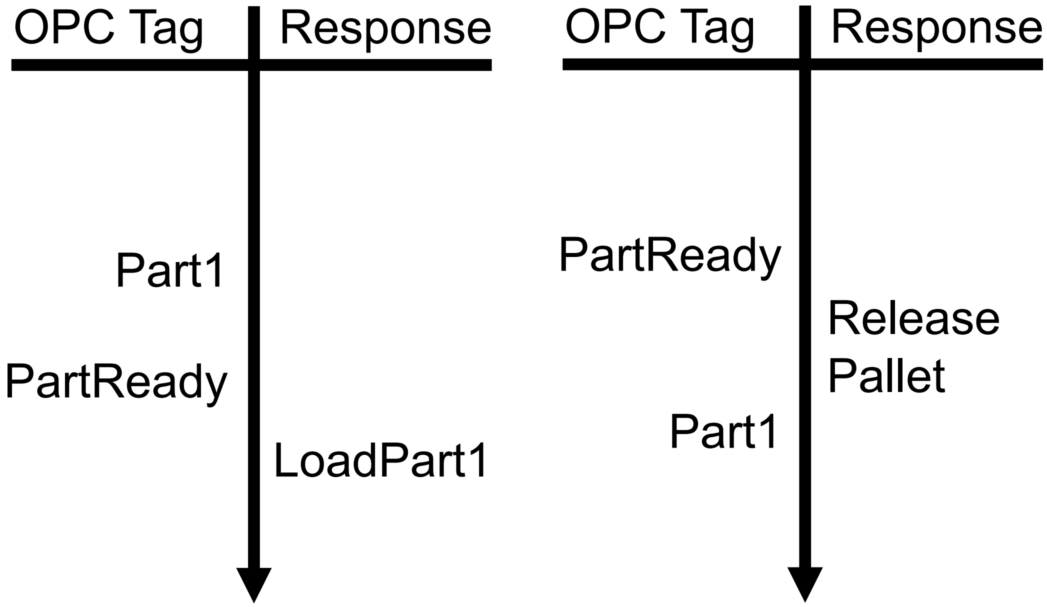
Guest Professor, Automatic Control, Lund University

Outline

- Motivating example
- Anomaly detection method
- Application to industrial system
- Conclusions and future work

Motivating example





Correct, typical behavior

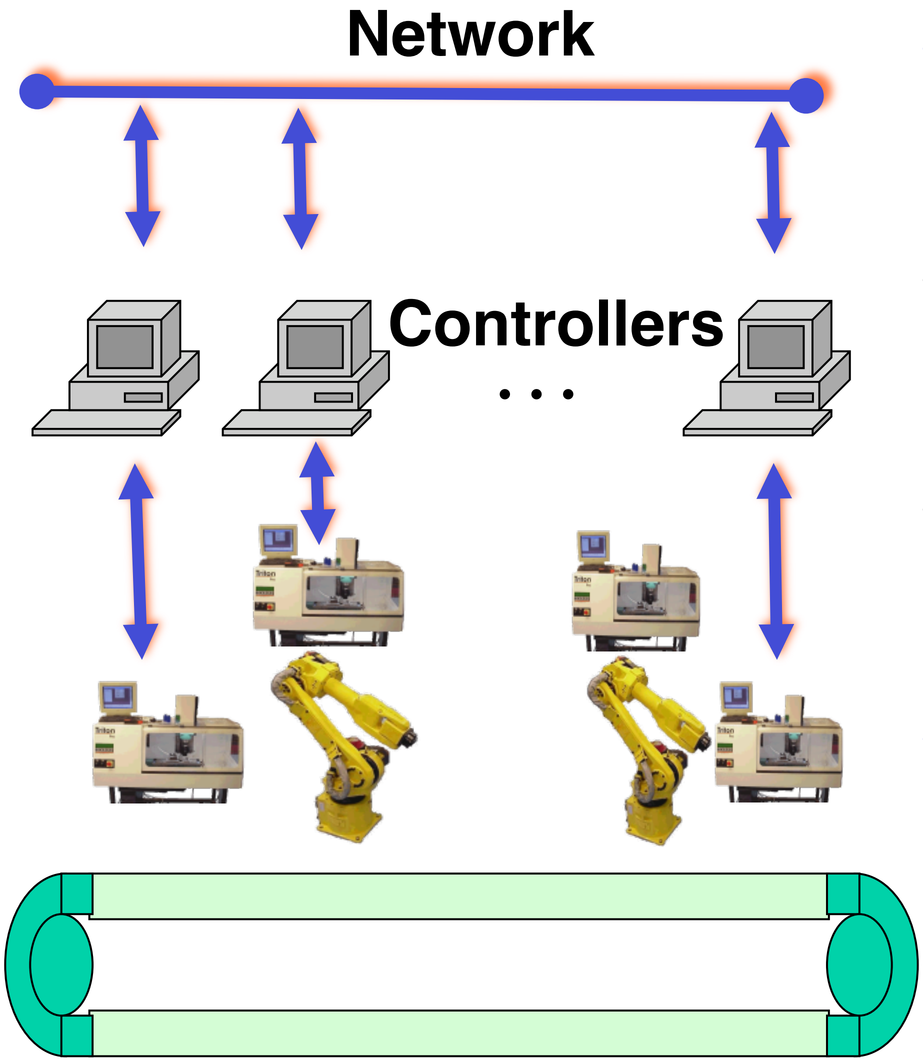
OPC Tag	Response
Part1	
PartReady	
	LoadPart1

Incorrect behavior

OPC Tag	Response
PartReady	
	Release Pallet
Part1	

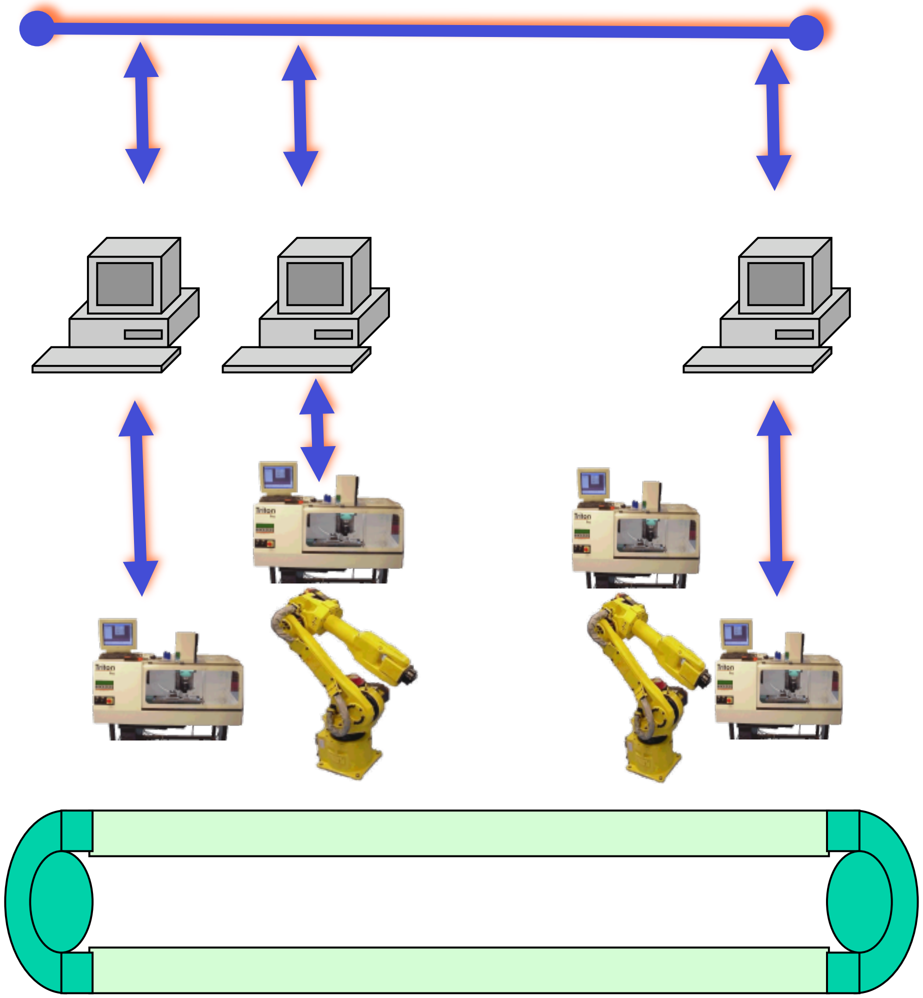
- **No model of entire system’s correct behavior**
- **Manual inspection to find the anomaly**
  - **a laborious, offline process**

Manufacturing systems



- Resource
  - Robot, machine, conveyor, pallet …
- Controller
  - machine control, system-level, PLC, …
- Communication
  - Networks carry events between controllers
- Process
  - Set of disjoint events
  - Use shared resources to accomplish a goal

Problem statement



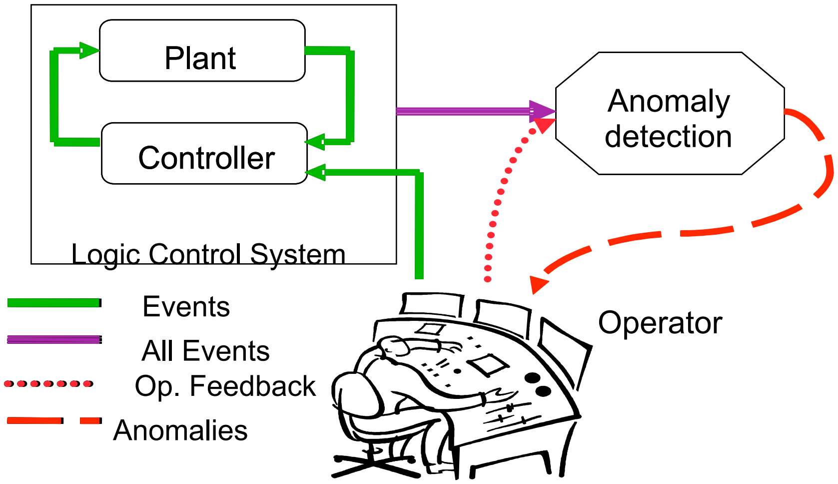
- Given a manufacturing system with known resources and processes, and a record of events sent between controllers,
- Find anomalies in the event-based communication records

Outline

- Motivating example
- Anomaly detection method
  - Assumptions and Petri net formalism
  - Model generation
  - Model performance assessment
  - Anomaly detection
- Application to industrial system
- Conclusions and future work

Goal: Anomaly detection

- **Assumptions:**
  - No formal model
  - Resources are known
  - Processes are known
  - Events are recorded
    - Events associated with resources and processes



Plant

Controller

Logic Control System

Anomaly detection

Operator

Events

All Events

Op. Feedback

Anomalies

- **Method:**
  - Generate models based on training data
  - Detect anomaly by comparing trace to models
  - Advise the operator when anomaly occurs

Assumptions

- **Known:**
  - Resources and capacities
    - Robots
    - CNC machines
    - Pallets
  - Events that acquire & release resources
- **Measurable:**
  - Event logs
    - Communication between controllers (OPC tag changes)
- **Unknown:**
  - Formal model of the system
    Could be constructed but is time-consuming and error prone
  - Logic control code
    Written by different people at different times in different languages
  - Correct event order
    Many different orders may be acceptable

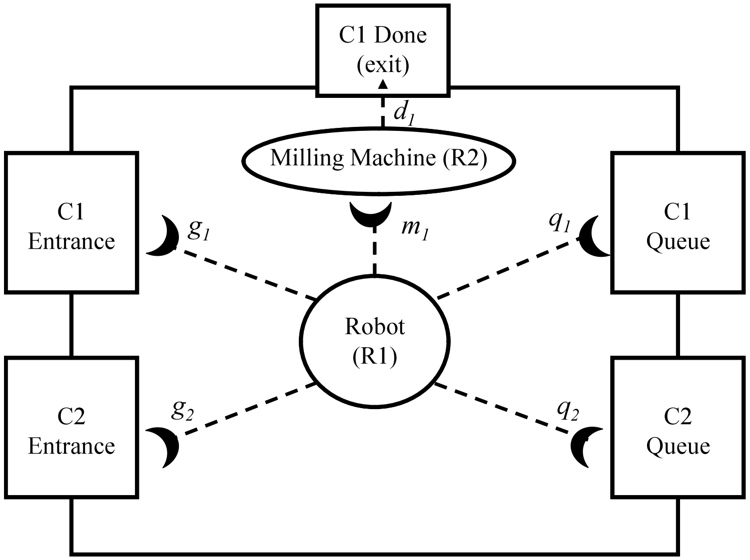
Prior work on fault detection

Existing approaches	Our approach
Machine-level faults	System-level faults
Continuous data	Event-based data
Require pre-existing formal model	No pre-existing formal model
Require fault models	No model of faults
Require system knowledge, e.g. max # of places in Petri net	No *a priori* limit on size of system model
No decisions based on resource availability	Readily-available system information included

SPSR = System of Processes that interact through Shared Resources

- Resource – robot, CNC, pallet, etc.
  - Acquired and released by known events
  - Capacity of each resource is known
- Process = set of events and resources
  - Processes interact to accomplish a goal
  - Modular (separate, independent)
  - Interact only through shared resources

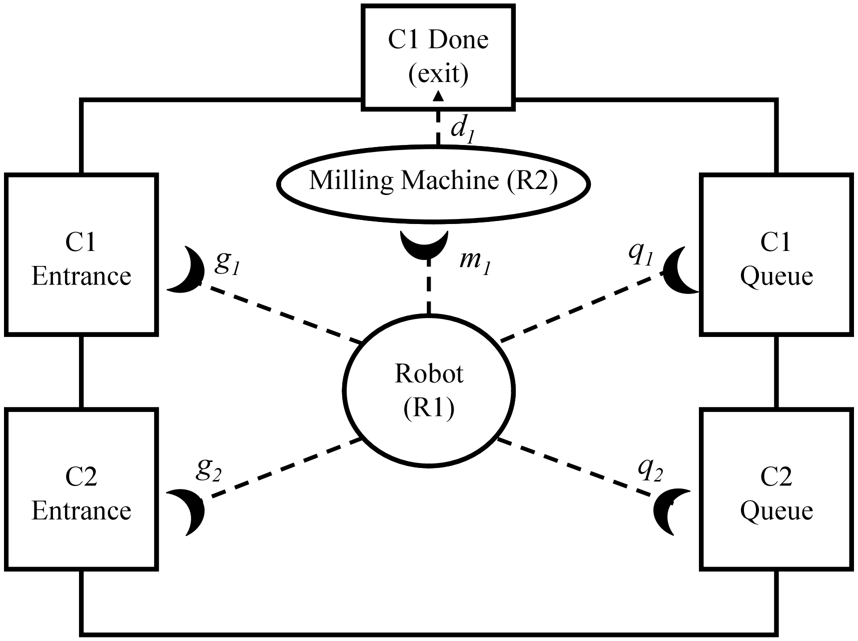
Example: Resources and events

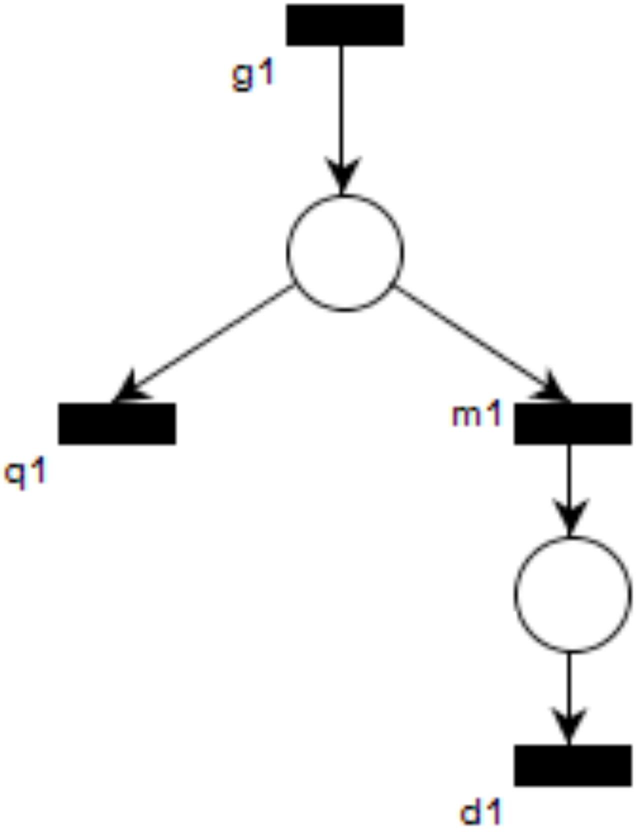


	Acquire	Release
R1 (Robot)	g_1, g_2	m_1, q_1, q_2
R2 (Machine)	m_1	d_1

- **Process 1**
  - **Events g_1, q_1, m_1, d_1**
- **Process 2**
  - **Events g_2, q_2**
- **Resources**
  - **Robot; shared by both processes**
  - **Milling machine, only for process 1**

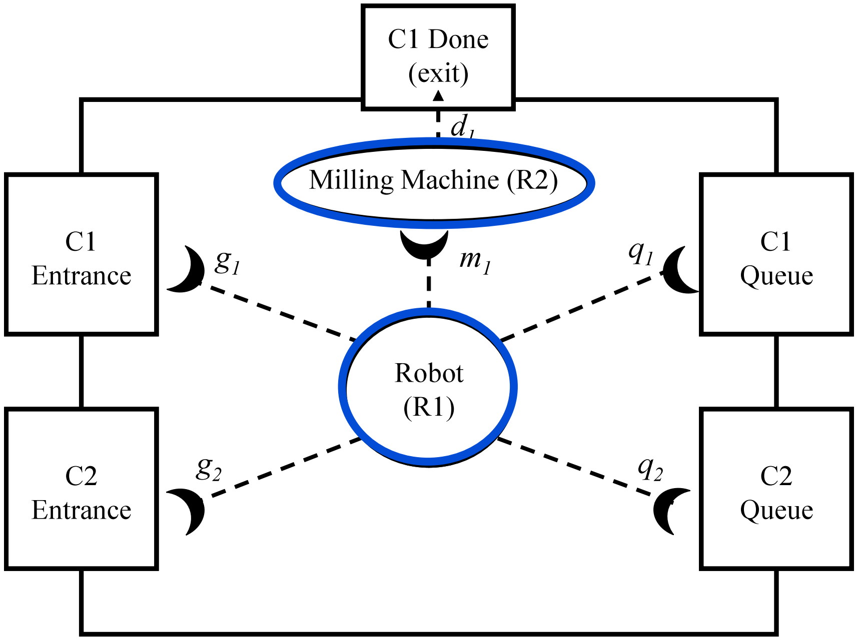
Transition Process (TP)

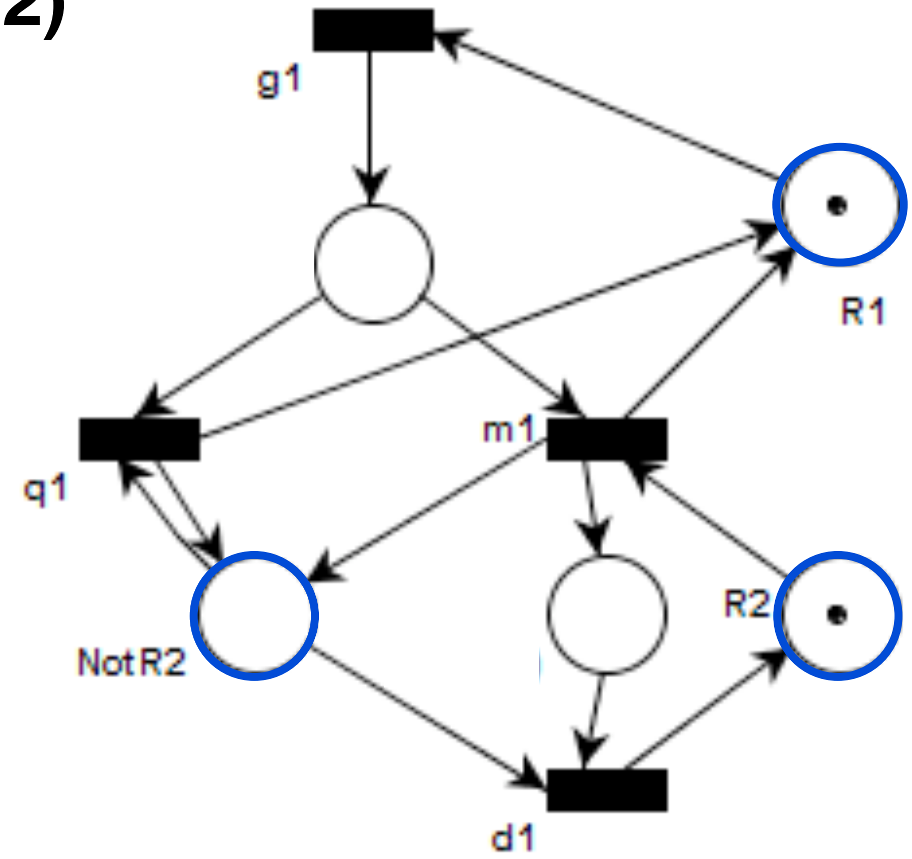




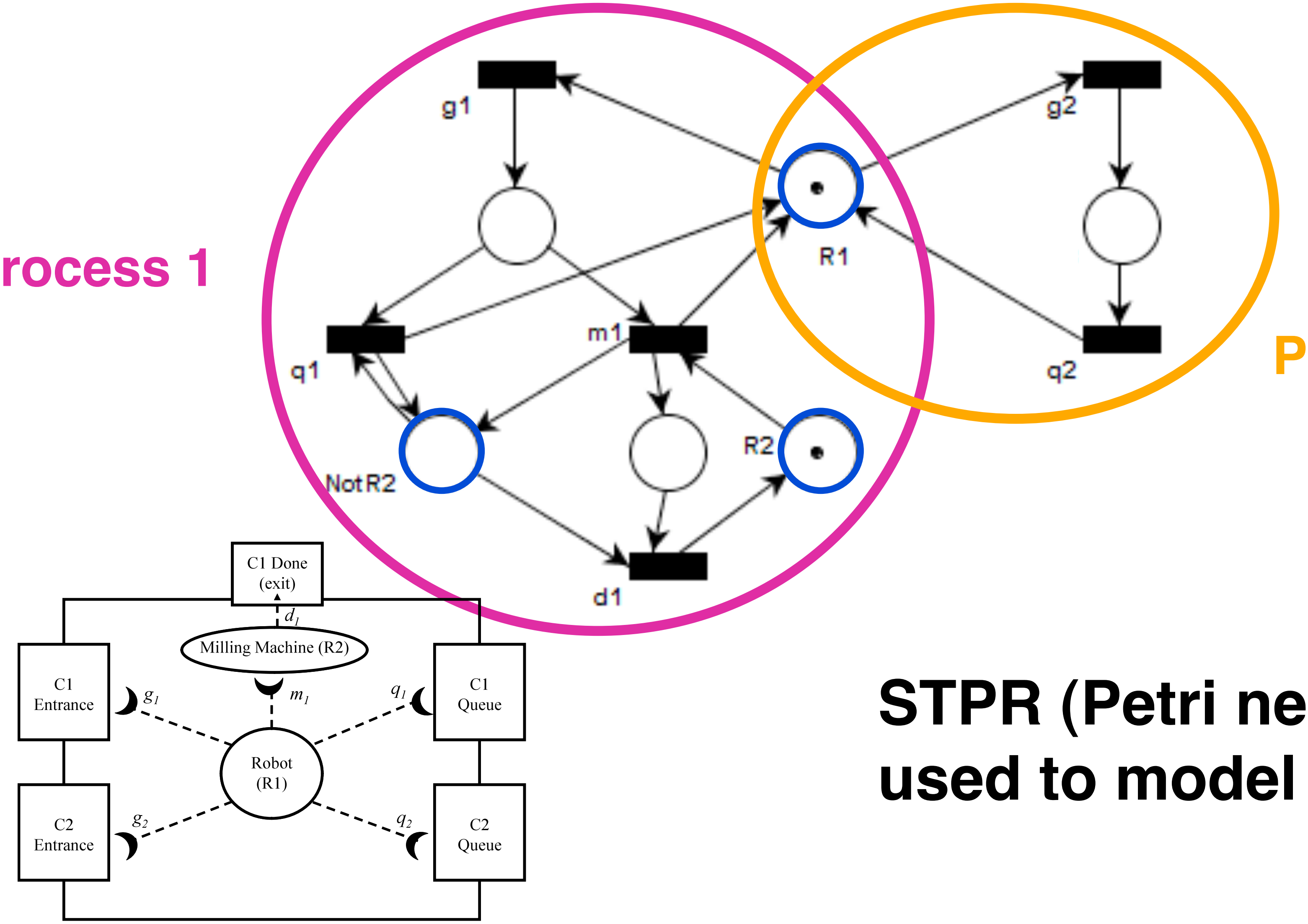
Transition Process with Resources

- **Decision to mill or queue based on availability of mill (R2)**





Systems of Transition Processes with Resources (STPRs)



Process 1

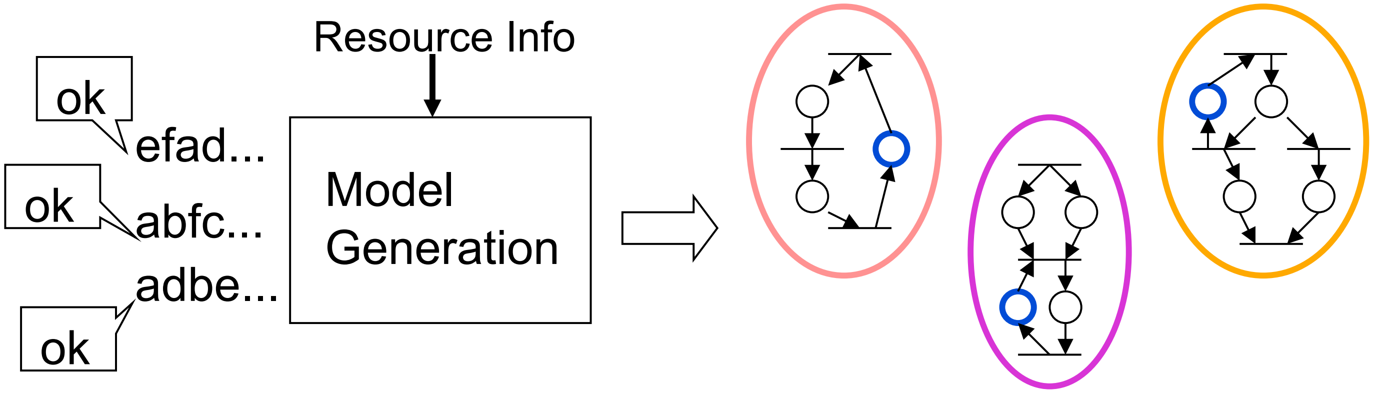
Process 2

STPR (Petri net) used to model SPSR

Outline

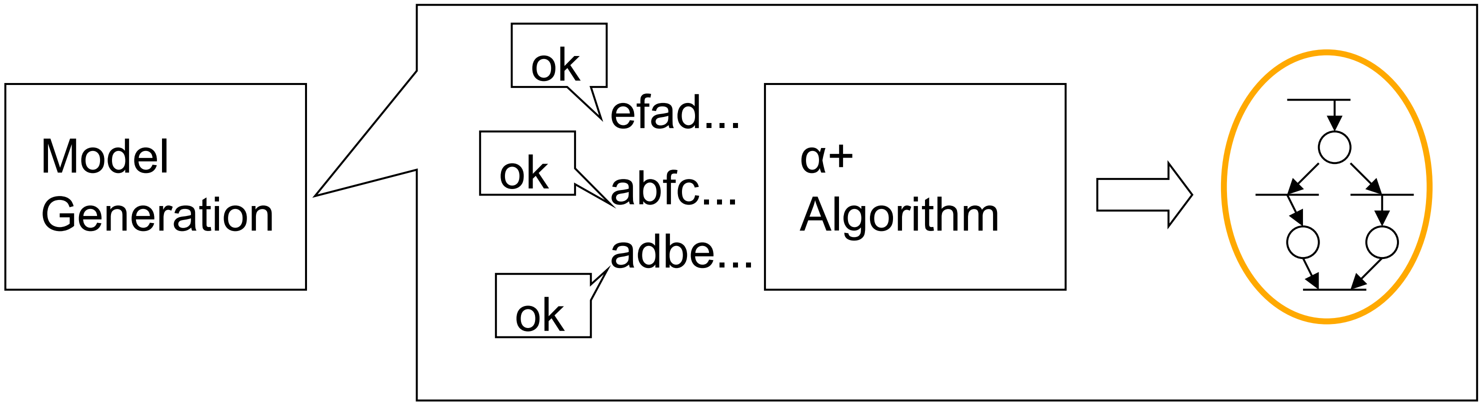
- Motivating example
- Anomaly detection method
  - Assumptions and Petri net formalism
  - Model generation
  - Model performance assessment
  - Anomaly detection
- Application to industrial system
- Conclusions and future work

Model Generation



- **Input: normal event streams, resource info**
- **Create models of each process**
- **Connect process models via shared resources**
- **Output: models of whole system**

Create individual process models



- **Determine process-specific event relationships**
- **Create process models (α+ algorithm)**
  - **Variations considering other process events**
  - **Add resource information to models**

α+ algorithm for process models

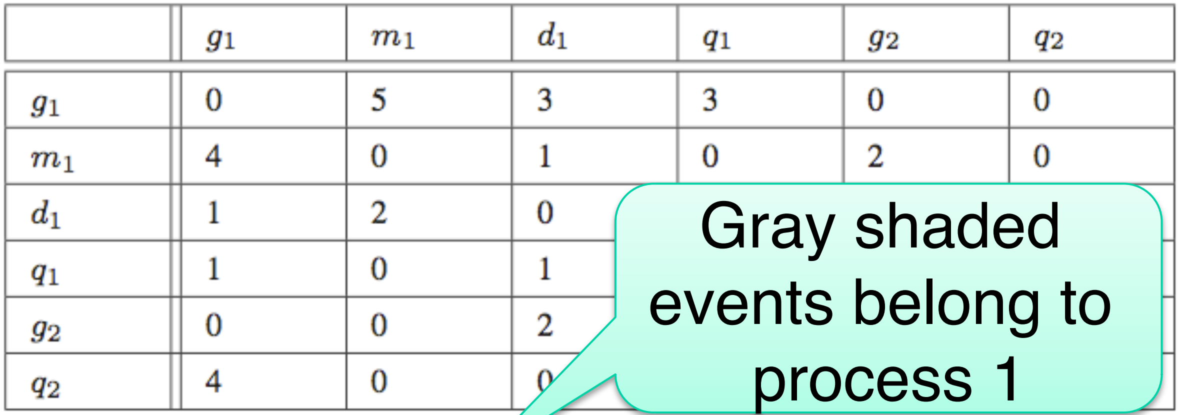
- Event ordering relationships
  - Causal if ab but not ba (→)
  - Two-event loop if both aba and bab occur (◊)
  - Parallel if both ab and ba occur (II)
  - None if neither ab nor ba occurs (#)
- Creating places (events label transitions)
  - Causal: One place from a to b
  - Loop: Two places connecting a and b
  - Parallel or None: No places created

Event pair relationships

$\sigma_1 = g_1 m_1 g_1 d_1 m_1 d_1 g_1 m_1 g_1 q_1 g_1 q_1 g_2 d_1 q_2 g_1 m_1 g_2 d_1 q_2$

$\sigma_2 = g_2 q_2 g_1 m_1 g_1 q_1 d_1 g_2 q_2 g_2 q_2 g_1 m_1 g_2 q_2 g_1 d_1 m_1 g_1 d_1$

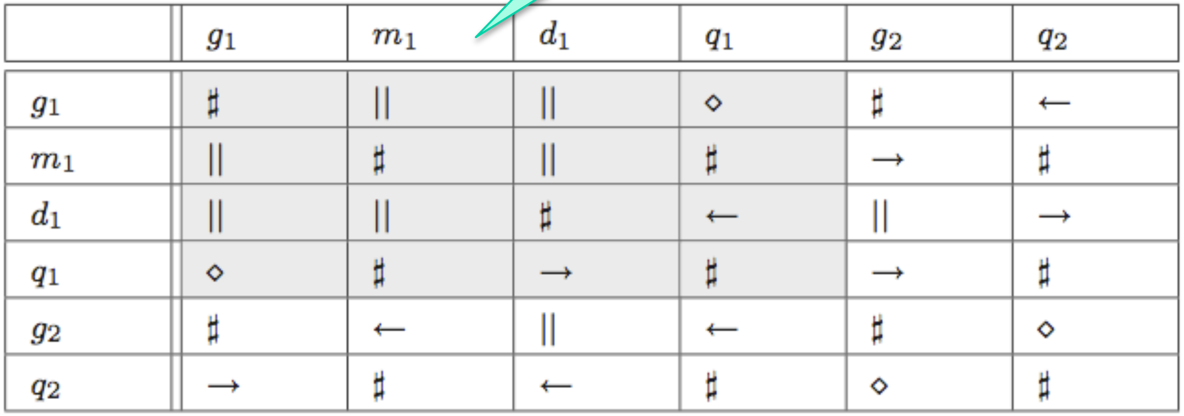
- **Number of occurrences of each pair**



	g_1	m_1	d_1	q_1	g_2	q_2
g_1	0	5	3	3	0	0
m_1	4	0	1	0	2	0
d_1	1	2	0	[illegible]	[illegible]	[illegible]
q_1	1	0	1	[illegible]	[illegible]	[illegible]
g_2	0	0	2	[illegible]	[illegible]	[illegible]
q_2	4	0	0	[illegible]	[illegible]	[illegible]

Gray shaded events belong to process 1

- **Relationships between pairs**



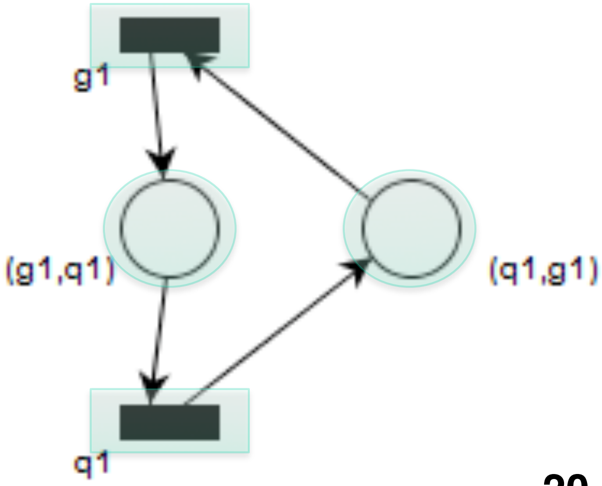
	g_1	m_1	d_1	q_1	g_2	q_2
g_1	$\sharp$	$\|\|$	$\|\|$	$\diamond$	$\sharp$	$\leftarrow$
m_1	$\|\|$	$\sharp$	$\|\|$	$\sharp$	$\rightarrow$	$\sharp$
d_1	$\|\|$	$\|\|$	$\sharp$	$\leftarrow$	$\|\|$	$\rightarrow$
q_1	$\diamond$	$\sharp$	$\rightarrow$	$\sharp$	$\rightarrow$	$\sharp$
g_2	$\sharp$	$\leftarrow$	$\|\|$	$\leftarrow$	$\sharp$	$\diamond$
q_2	$\rightarrow$	$\sharp$	$\leftarrow$	$\sharp$	$\diamond$	$\sharp$

Example pair: 2-event loop

$\sigma_1 = g_1 m_1 g_1 d_1 m_1 d_1 g_1 m_1 g_1 q_1 g_1 q_1 g_2 d_1 q_2 g_1 m_1 g_2 d_1 q_2$

$\sigma_2 = g_2 q_2 g_1 m_1 g_1 q_1 d_1 g_2 q_2 g_2 q_2 g_1 m_1 g_2 q_2 g_1 d_1 m_1 g_1 d_1$

- $g_1 q_1 g_1$ and $q_1 g_1 q_1$ both occur
- $g_1 \lozenge q_1$ (two-event loop)
- One place to connect g_1 to q_1 and another place to connect q_1 to g_1



Example pair: Parallel

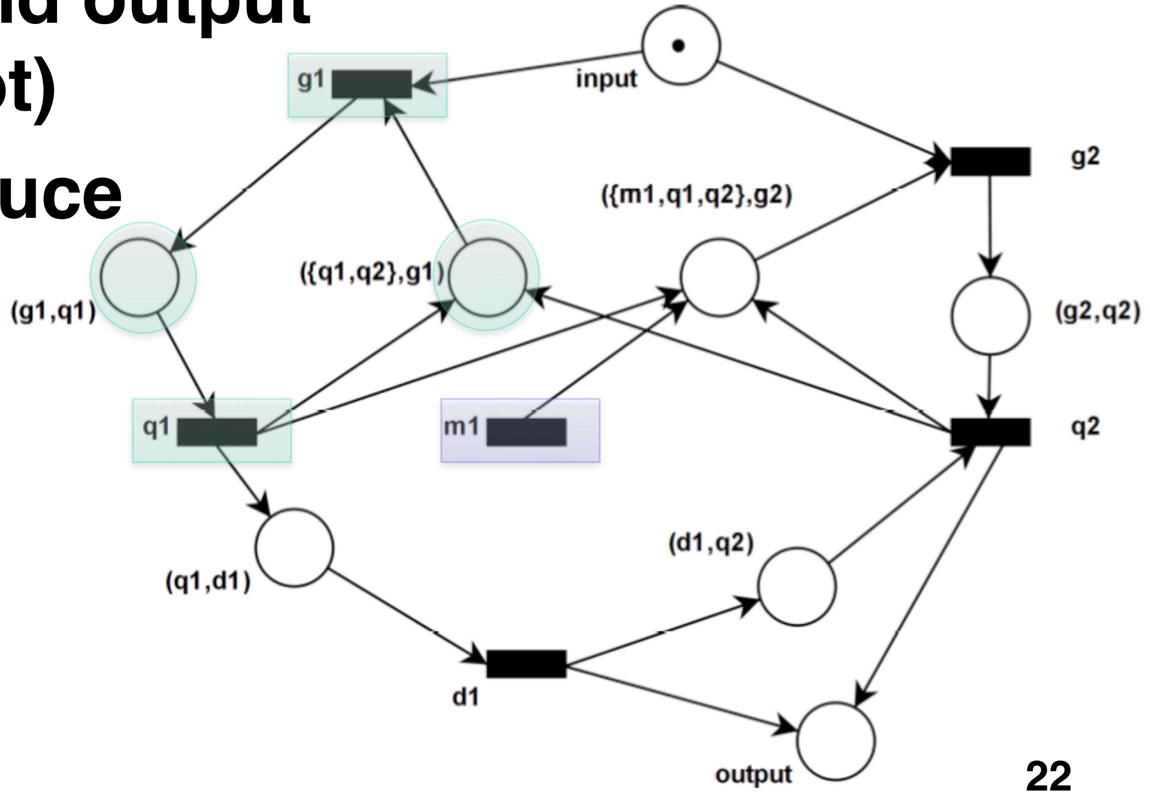
$\sigma_1 = g_1 m_1 g_1 d_1 m_1 d_1 g_1 m_1 g_1 q_1 g_1 q_1 g_2 d_1 q_2 g_1 m_1 g_2 d_1 q_2$

$\sigma_2 = g_2 q_2 g_1 m_1 g_1 q_1 d_1 g_2 q_2 g_2 q_2 g_1 m_1 g_2 q_2 g_1 d_1 m_1 g_1 d_1$

- $g_1 m_1$ Yes $m_1 g_1$ Yes $g_1 m_1 g_1$ Yes $m_1 g_1 m_1$ No
- $g_1 \mid\mid m_1$ (parallel)
- no places connecting

Output of α+ algorithm

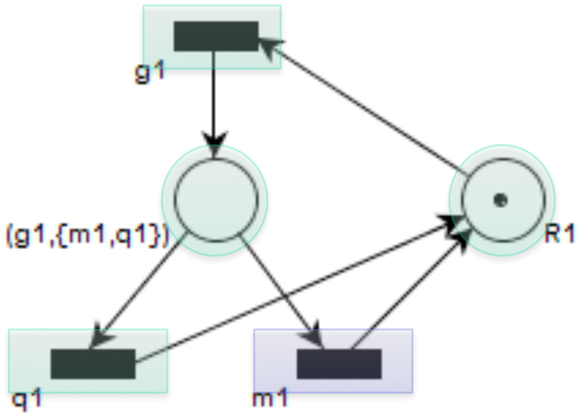
- Places created with → and ◊ relationships
- Combining places where possible
- Single input and output place (one-shot)
- Cannot reproduce given traces
- Does not incorporate resource info



Model variations

- Using only events from process 1 creates a single model for process 1
- Alternate models are created by:
  - Considering events from other processes
  - Considering resource relationships
  - Considering implicit event relationships (e.g., due to interleavings)
- Many variations created for each process
  - Multiple models for each process
- Process models are combined by joining shared resource places

Example: Incorporating resources



- g_1 || m_1 (no places connecting)
- Consider resource R1 (robot)
  - g_1 acquires R1
  - Both m_1, q_1 release R1
  - Subtract resource relationship
- g_1 || m_1 remove $m_1 \rightarrow g_1$ yields $g_1 \rightarrow m_1$
  - Add a place connecting g_1 to m_1
  - Combine places

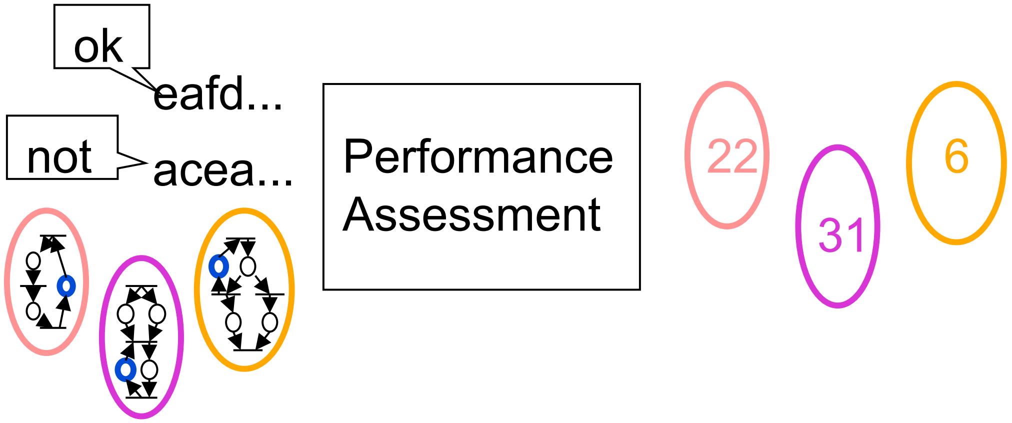
Theorem: Re-create “true” model

- **If** the underlying “true” model
  - is a TP with certain properties (safe, live, etc.)
  - and **if** the given event log contains all possible event pairs and two-event loops
- **Then** one of the TP models created by the model generation algorithm will be the “true” TP model.
- **Implication:** If the underlying “true” model is an STPR whose TPs and event log meet these requirements, then one of the created STPR models exactly matches the “true” model.

Outline

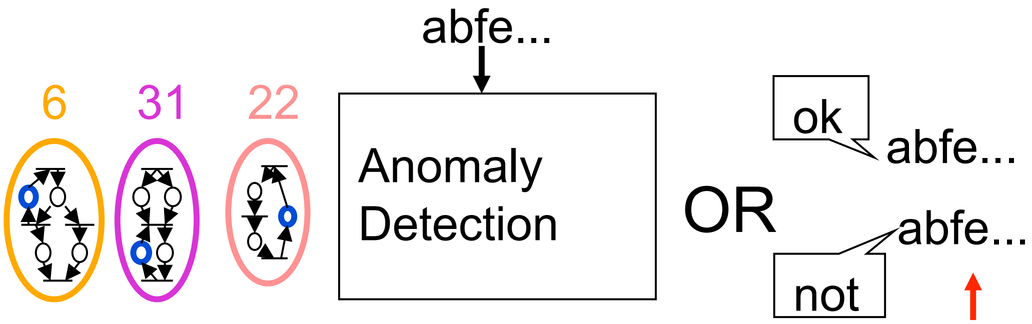
- Motivating example
- Anomaly detection method
  - Assumptions and Petri net formalism
  - Model generation
  - Model performance assessment
  - Anomaly detection
- Application to industrial system
- Conclusions and future work

Model Performance Assessment



- **Input: models, labeled event streams**
- **Assess model performance based on labeled event streams**
  - **Increase performance by 1 if correct**
  - **Decrease performance by 1 if incorrect**
- **Output: performance of models**

Anomaly Detection



- **Input: models with their performances, unlabeled event stream**
- **Anomaly detection**
  - **Determine whether models allow stream**
  - **Performance-weighted majority voting to decide whether anomaly in stream**
- **Output: whether anomaly in stream, and if so, where first detected**

Anomaly detect

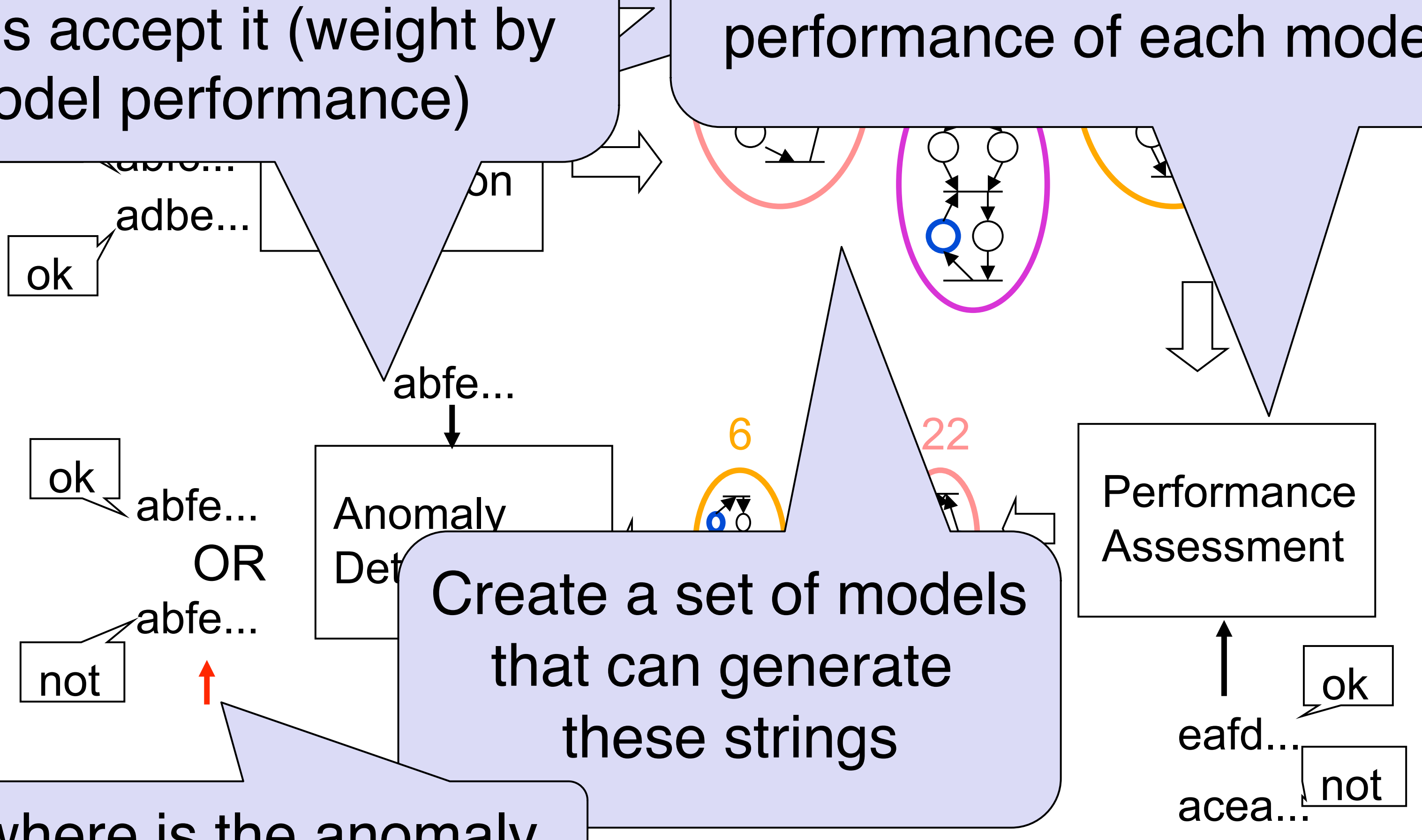
Given resource
nfor

Given a new string, determine whether the models accept it (weight by model performance)

Given some “ok” and “not ok” strings, compute the performance of each model

Create a set of models that can generate these strings

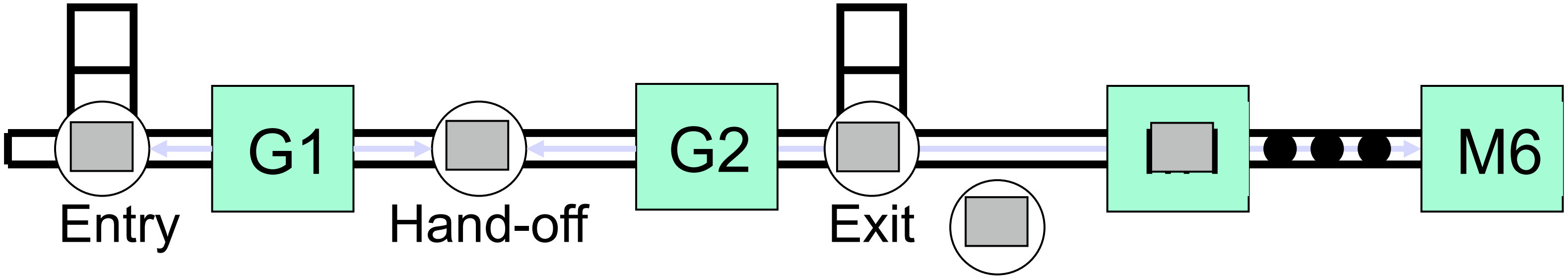
If not, where is the anomaly



Outline

- Motivating example
- Anomaly detection method
- Application to industrial system
- Conclusions and future work

Machining Cell: Physical Set-Up

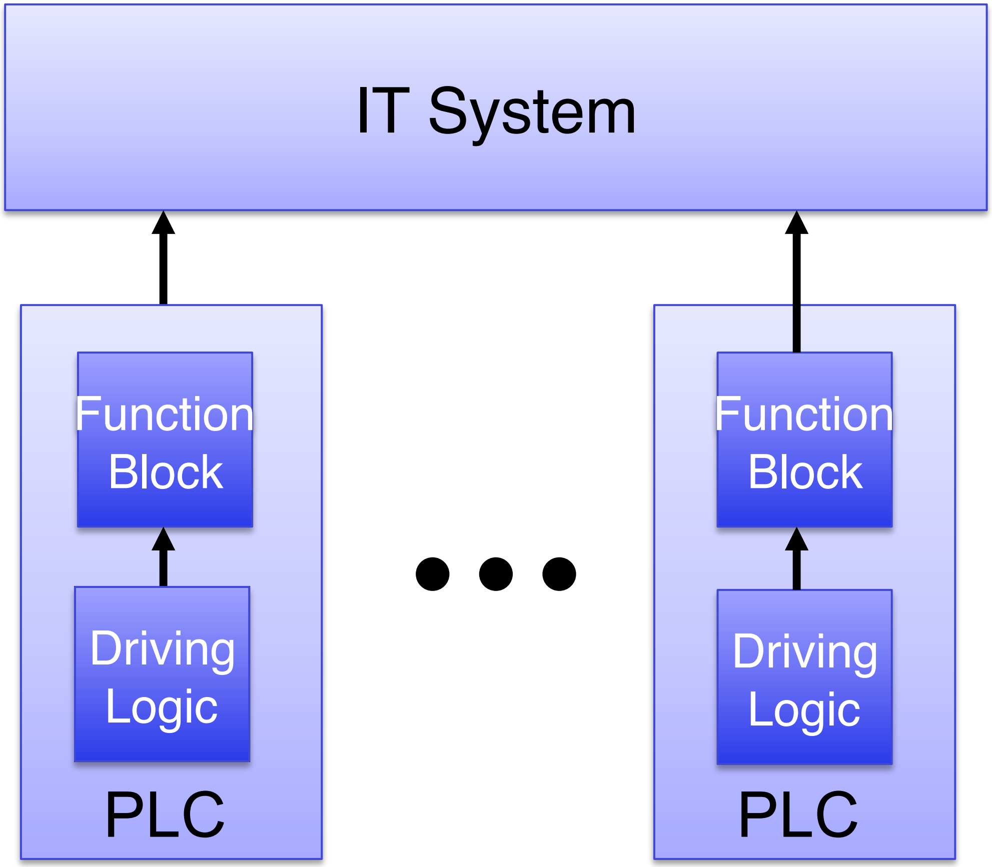


Reject

- **Problem: G2 will have raw parts and at least one CNC available, but G2 incorrectly waits**
- **Resources:**
  - **gantries, Machines (CNC)**
- **Processes:**
  - **one per CNC, one for gantries**

Data collection set-up

- Data from each machine & gantry
  - Bits include: Cycle End, Good/Bad Cycle, Wait Aux, Blocked, and Starved
  - PLC message generated each time particular bit changes occur



- Approx. 11,000 parts worth of data (270,000 PLC messages)

Application to industry data

- **What we thought we would get:**
  - **Well-defined strings of events**
  - **Events that acquire/release resources recorded**
  - **Unique mapping of PLC bits to events**
  - **Many strings, starting from the initial state, labeled as “good” or “bad**
- **What we got:**
  - **Not every event triggers a message → multi-bit change (order is uncertain)**
  - **Not all resource events captured in data collection**
  - **Some bits used for multiple purposes**
  - **One huge Excel file with no defined “beginning”**

Identified Inconsistencies

	Academic Assumptions	Industry Realities
1	Resource events available	Some events filtered in data collection
2	String of ordered events	Multiple bit changes per message possible
3	Consistent bit-meaning mapping	Inconsistent bit-meaning mapping
4	Event streams start in initial state	System runs continuously
5	Separate, labeled streams	Continuous, unlabeled stream

1) Acquire/Release Resources

- Events that acquire and release resources are required for model-building
- Not all such events were recorded in the data collection system
- Potential solution: proxy events
  - Example: gantry picks up raw parts
    - Proxy: gantry begins unload/load CNC
    - Problem: Do not know when gantry is waiting
- Actual solution: Industry changed data collection system to record these events

2) String of Ordered Events

- Ordering of events not known; multiple bit changes (MBC) between PLC messages
- Possible causes of MBC
  - Not all bit changes cause PLC messages
  - Multiple bits can change within one PLC scan
- Potential solution: Treat each MBC as unique event
  - Approximately 1/3 of messages are MBC
  - MBCs account for > 5/6 unique events
- Actual solution: Split each MBC into sequence of single events

3) Consistent bit mapping

- Design documents define meaning of bits
- Implementation of PLC programming may result in slightly different use of bits
- Examples (occasional, inconsistent)
  - Cycle End bit pulsed high twice in a row
  - Wait Aux used for other other purposes besides machine interaction
- Potential solution: change logic to be consistent with design docs
- Actual solution: some logic changes, also pre-processed data for known issues

4) Initial State

- System is running continuously, rarely returns to “initial” state
- Problem: given event stream and STPR model, determine whether there exists a sequence of states in the model such that event stream could have occurred
- Solution: Define a necessary condition
    - Lower bound based on events in stream
    - Upper bound based on resource conservation
    - If lower bound < upper bound, stream is possible

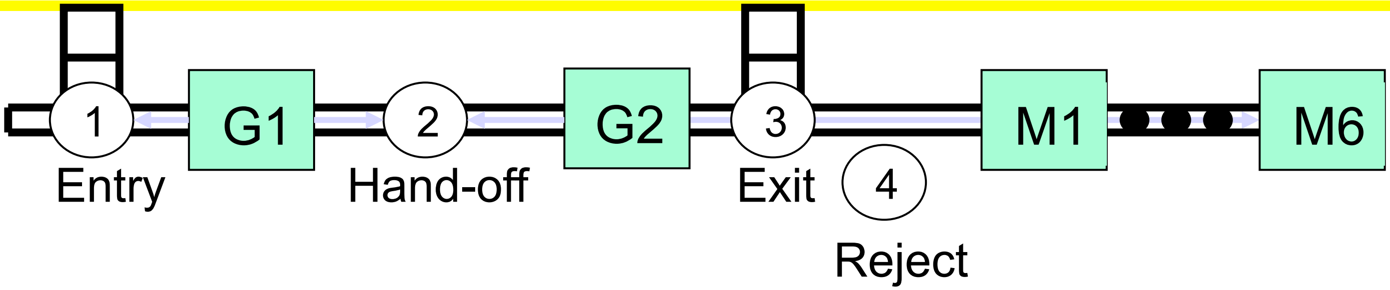
5) Separate Labeled Streams

- **Labeled streams needed**
  - **Normal to create models**
  - **Normal & anomalous to assess model performance**
- **Potential solution: System expert adds labels to streams**
- **Actual solution: Algorithm to split and label streams**
  - **Split data into pre-set size streams**
  - **Label streams based on conditions on events that are know to be associated with problem(s)**

Inconsistencies & Resolutions

	Academic Assumptions	Industry Realities	Resolution
1	Resource events available	Some events filtered in data collection	**I**: Logic changed
2	String of ordered events	Multiple bit changes per message possible	**A**: Heuristic decision algorithm
3	Consistent bit-meaning mapping	Inconsistent bit-meaning mapping	**I, A**: Logic changed, pre-process data
4	Event streams start in initial state	System runs continuously	**A**: Nec. condition to create stream
5	Separate, labeled streams	Continuous, unlabeled stream	**A**: Splitting, labeling algorithm

Machining cell: Data selection



- 8 PLCs each report 40 words (x16 bits) data
  - Appropriate events (bit rise/fall) must be chosen

Resource	Events	Resource	Events
Gantry	Start waiting	CNCi	Blocked
	End waiting		Wait Aux
	Leave w/o waiting		Part not present
	Pick up raw part		Cycle end
	Part at inspection		
	Part at exit		
	Arrive at CNCi		

Building model from data

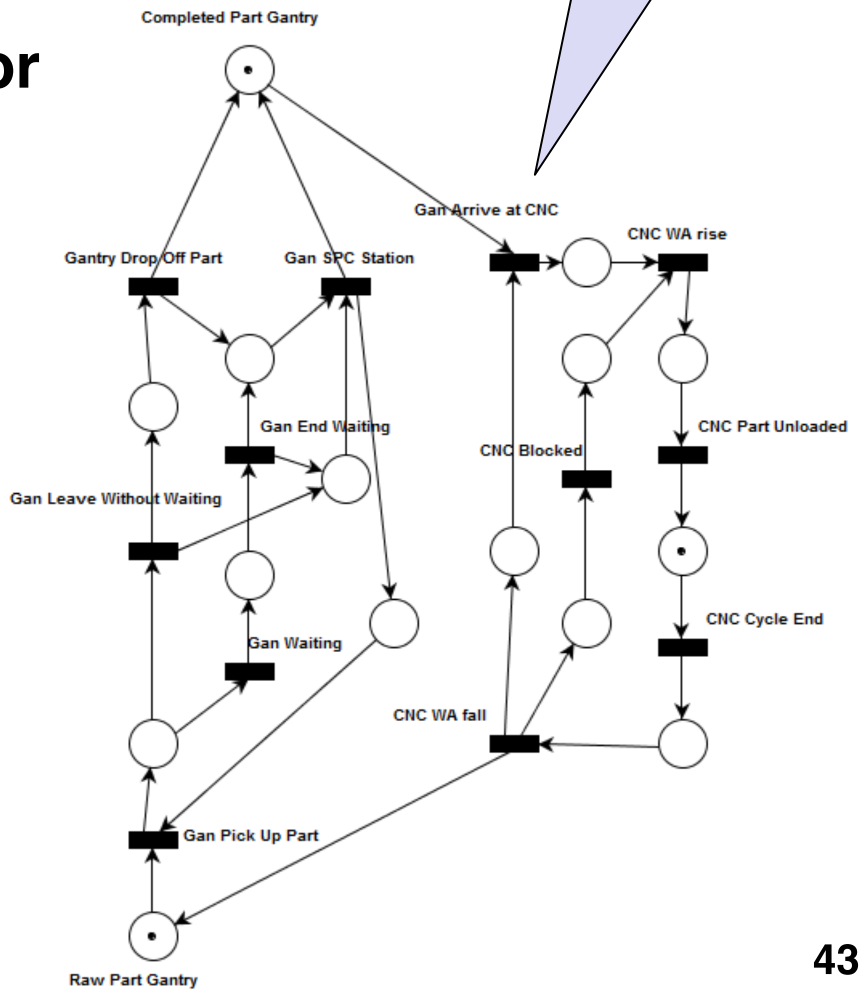
- One process for gantry, one for each CNC
- Exclude sections of data when gantry or CNC in “fault” state
- Add *CNCs Ready* resource as combination of all CNCs

Resources	Acquire	Release
Gantry G1	Pick up raw part	CNC wait aux fall
Gantry G2	Arrive at CNCi	Part at exit; inspection
Mi (CNCi)	CNC cycle end rise	CNC part unload
CNCs Ready	Gantry end waiting	CNC blocked

Anomaly detection results

Sample model with one CNC process

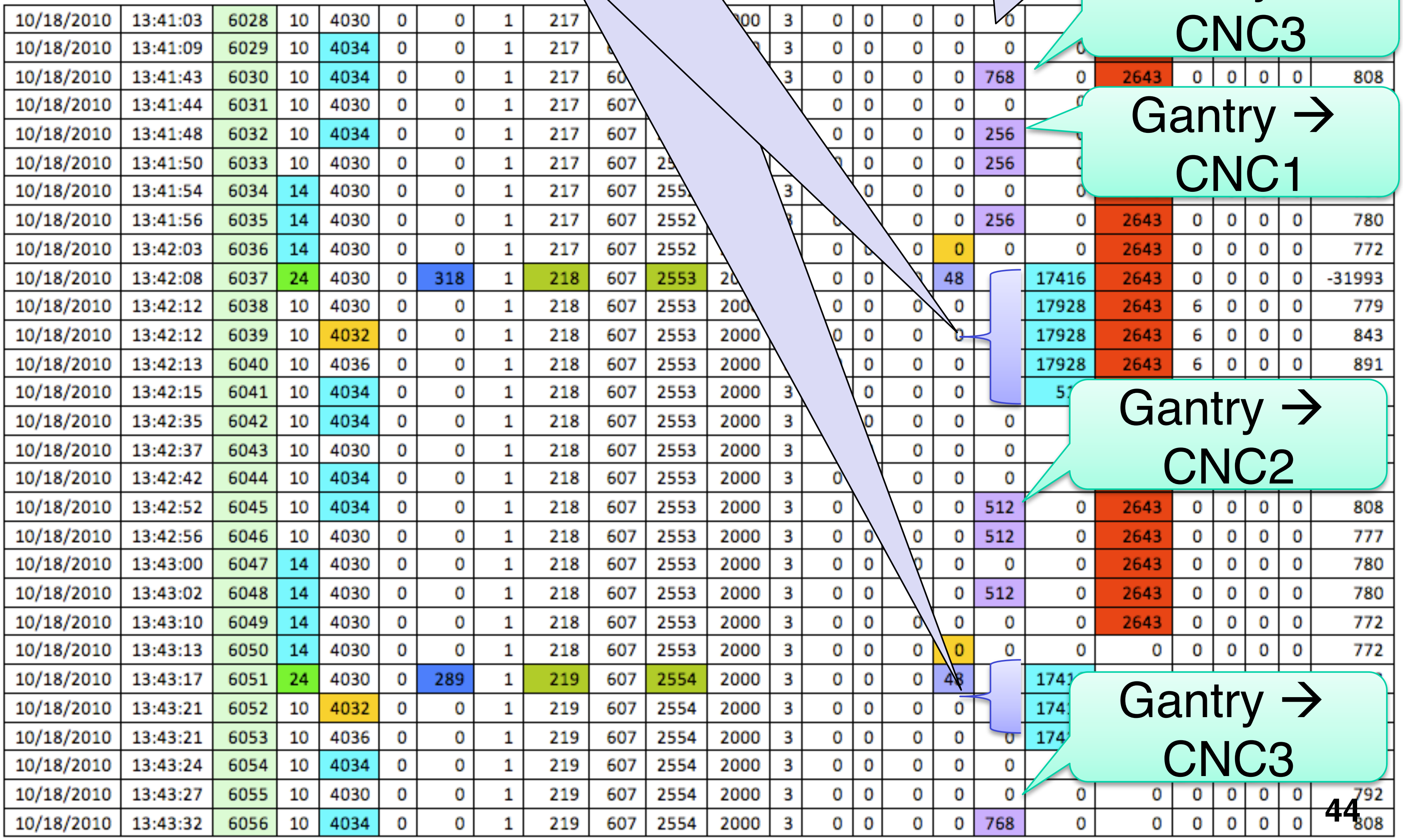
- Use part of data to build models
  - Other part to check for anomalies
  - Repeat with different subsets
- Anomalies detected:
  - Gantry waiting while CNC blocked
  - Part dropped off at inspection station



Indu

Gantry waiting: word 19 bit 9 is high

Word 18 bits 8-10 give the CNC ID



Gantry → CNC3

Gantry → CNC1

Gantry → CNC2

Gantry → CNC3

10/18/2010	13:41:03	6028	10	4030	0	0	1	217	[illegible]	[illegible]	[illegible]	3	0	0	0	0	0	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
10/18/2010	13:41:09	6029	10	4034	0	0	1	217	[illegible]	[illegible]	[illegible]	3	0	0	0	0	0	0	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
10/18/2010	13:41:43	6030	10	4034	0	0	1	217	[illegible]	[illegible]	[illegible]	3	0	0	0	0	768	0	2643	0	0	0	0	808
10/18/2010	13:41:44	6031	10	4030	0	0	1	217	607	[illegible]	[illegible]	[illegible]	0	0	0	0	0	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
10/18/2010	13:41:48	6032	10	4034	0	0	1	217	607	[illegible]	[illegible]	[illegible]	0	0	0	0	256	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
10/18/2010	13:41:50	6033	10	4030	0	0	1	217	607	[illegible]	[illegible]	[illegible]	[illegible]	0	0	0	256	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
10/18/2010	13:41:54	6034	14	4030	0	0	1	217	607	[illegible]	[illegible]	3	[illegible]	0	0	0	0	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
10/18/2010	13:41:56	6035	14	4030	0	0	1	217	607	2552	[illegible]	[illegible]	[illegible]	[illegible]	0	0	256	0	2643	0	0	0	0	780
10/18/2010	13:42:03	6036	14	4030	0	0	1	217	607	2552	[illegible]	[illegible]	0	[illegible]	0	0	0	0	2643	0	0	0	0	772
10/18/2010	13:42:08	6037	24	4030	0	318	1	218	607	2553	[illegible]	[illegible]	0	0	[illegible]	48	[illegible]	17416	2643	0	0	0	0	-31993
10/18/2010	13:42:12	6038	10	4030	0	0	1	218	607	2553	[illegible]	[illegible]	0	0	0	0	[illegible]	17928	2643	6	0	0	0	779
10/18/2010	13:42:12	6039	10	4032	0	0	1	218	607	2553	2000	[illegible]	[illegible]	0	0	0	[illegible]	17928	2643	6	0	0	0	843
10/18/2010	13:42:13	6040	10	4036	0	0	1	218	607	2553	2000	[illegible]	[illegible]	0	0	0	[illegible]	17928	2643	6	0	0	0	891
10/18/2010	13:42:15	6041	10	4034	0	0	1	218	607	2553	2000	3	[illegible]	0	0	0	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
10/18/2010	13:42:35	6042	10	4034	0	0	1	218	607	2553	2000	3	[illegible]	[illegible]	0	0	0	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
10/18/2010	13:42:37	6043	10	4030	0	0	1	218	607	2553	2000	3	[illegible]	[illegible]	0	0	0	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
10/18/2010	13:42:42	6044	10	4034	0	0	1	218	607	2553	2000	3	0	[illegible]	0	0	0	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
10/18/2010	13:42:52	6045	10	4034	0	0	1	218	607	2553	2000	3	0	[illegible]	0	0	512	0	2643	0	0	0	0	808
10/18/2010	13:42:56	6046	10	4030	0	0	1	218	607	2553	2000	3	0	0	[illegible]	0	512	0	2643	0	0	0	0	777
10/18/2010	13:43:00	6047	14	4030	0	0	1	218	607	2553	2000	3	0	0	[illegible]	0	0	0	2643	0	0	0	0	780
10/18/2010	13:43:02	6048	14	4030	0	0	1	218	607	2553	2000	3	0	0	[illegible]	0	512	0	2643	0	0	0	0	780
10/18/2010	13:43:10	6049	14	4030	0	0	1	218	607	2553	2000	3	0	0	[illegible]	0	0	0	2643	0	0	0	0	772
10/18/2010	13:43:13	6050	14	4030	0	0	1	218	607	2553	2000	3	0	0	0	0	0	0	0	0	0	0	0	772
10/18/2010	13:43:17	6051	24	4030	0	289	1	219	607	2554	2000	3	0	0	0	48	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
10/18/2010	13:43:21	6052	10	4032	0	0	1	219	607	2554	2000	3	0	0	0	0	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
10/18/2010	13:43:21	6053	10	4036	0	0	1	219	607	2554	2000	3	0	0	0	0	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
10/18/2010	13:43:24	6054	10	4034	0	0	1	219	607	2554	2000	3	0	0	0	0	0	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
10/18/2010	13:43:27	6055	10	4030	0	0	1	219	607	2554	2000	3	0	0	0	0	0	0	0	0	0	0	0	792
10/18/2010	13:43:32	6056	10	4034	0	0	1	219	607	2554	2000	3	0	0	0	0	768	0	0	0	0	0	0	808

Outline

- **Motivating example**
- **Anomaly detection method**
- **Application to industrial system**
- **Conclusions and future work**

Conclusions & Future Work

- Anomaly detection in event-based systems without a formal model
  - Information on resources, processes & events
- Multiple process models built to cope with uncertainty about true behavior of system
  - Number of “whole” models can be quite large
  - Maintaining modularity through performance assessment phase could reduce complexity
- Application to off-line data
  - On-line detection could be implemented
  - On-line model building could consider and evaluate multiple different sets of events

Acknowledgements

- NSF Engineering Research Center for Reconfigurable Manufacturing Systems at the University of Michigan
  - Prof. Yoram Koren, Director
- Lindsay Allen, former PhD student
- John Broderick, MS student
- Industry partners