Decidable Problems for Counter Systems

Day 1
Introduction to Counter Systems

Stéphane Demri

demri@lsv.ens-cachan.fr

LSV, ENS Cachan, CNRS, INRIA

ESSLLI 2010, Copenhagen, August 2010

What is in the course?

Analysis of Counter Systems
From Reachability to Temporal Logics

- Day 1: Introduction to counter systems
- Day 2: Linear-time temporal logics
- Day 3: Vector addition systems
- Day 4: Reversal-bounded counter automata
- Day 5: Model-checking counter systems

but also Presburger arithmetic, undecidability, computational complexity, etc.

What can you expect to learn?

- Presentation of numerous **classes of counter systems**.
- **Proof techniques** to decide reachability problems for infinite-state systems.
- **Temporal reasoning** on transition systems.

Background

1. Necessary background
   - Basics of first-order logic and temporal logics.
   - Basics of finite-state automata and formal languages.
   - Basics of complexity theory, decidability.
2. Optional background
   - Temporal logic LTL and automata-based approach.
   - Petri nets.
   - Familiarity with complexity classes NP, PSPACE, EXPSPACE etc.

Course material

- Lecture notes available on

  `www.lsv.ens-cachan.fr/~demri/esslli10-course.html`

- Slides available on a daily basis (made from the lecture notes).

- Do not hesitate to contact me during ESSLLI or to send me emails at `demri@lsv.ens-cachan.fr`.

Formal Verification

Verification at the heart of computer science

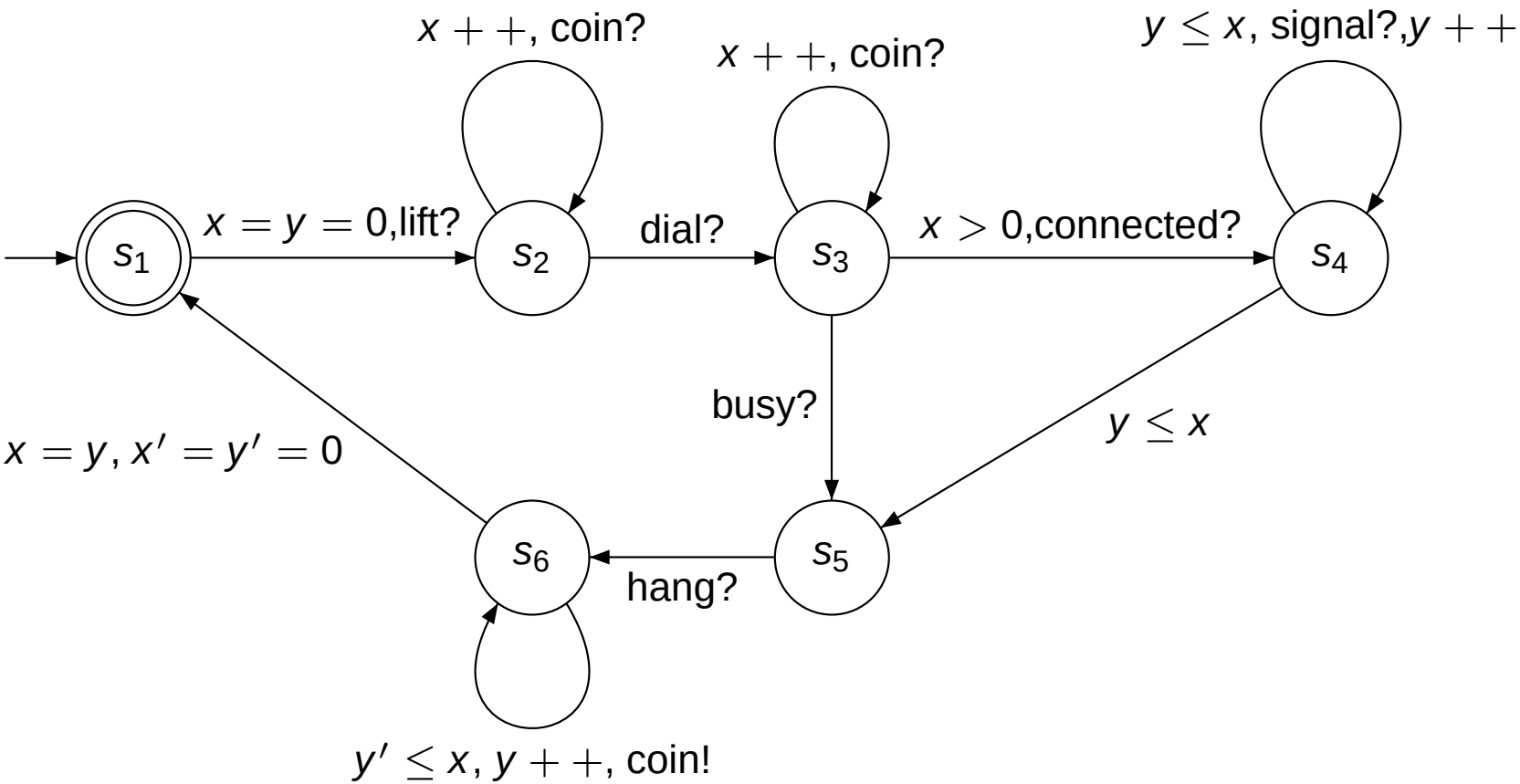
- Digital systems are everywhere.
  Desktops, embedded systems, cellular phones, etc.

- Needs for verifying functional/security properties:
  - Hardware components
  - Software (programs, communication protocols, web applications, . . .)

  Formal verification is a process in which mathematical techniques are used to guarantee the correctness of a design with respect to some specified behavior.
  [Halpern et al., BSL 01]

From systems to models

- Systems are modelled as abstract operational models (counter systems, timed automata, etc.).



Verification as a logical problem

- Properties are represented by logical formula.
  "The system $\mathcal{S}$ never reaches a bad state" $\longrightarrow$ $\mathtt{A}\,\mathtt{G}\,\neg\mathrm{bad}$.
- Logical problems involve abstract models and formulae.
- Development of procedures to solve these problems.
  automata, analytic proof systems, ad-hoc methods . . .
- Ultimate goal: automatic verification.
- There are theoretical limits for this entreprise.
  - The halting problem for Turing machines is undecidable.
    [Turing, 37]
  - The set of valid first-order formulae is undecidable.
    [Church, JSL 36]

Methodology

- System, property $\mapsto$ model, logical formula.
- Logical problems:
  - Decision problems (model-checking, validity, . . .)
  - Search problems (controller synthesis, query checking, . . .)
- Analysis of the computational resources to solve the problems
  - Decision procedures vs. undecidability.
  - Complexity in time or memory space.
- Classification
  - Generalizing the models or logics (e.g., Extended TL).
  - Fragments with better computational properties (e.g., FO2).
  - Variants such as fragments of generalizations (e.g., one-clock alternating timed automata).

Formal verification and temporal logics

- Aspects of temporality in computer science
  - Specification and verification of concurrent/reactive systems.
  - Real-time processes and systems.
  - Temporal databases.

- Logics as formal specification languages
  - To define mathematically the correctness of systems.
  - To express properties without ambiguities.
  - To make formal proofs and develop generic methods.

Model-checking and temporal logic

- Temporal logic for specifying behaviors of reactive systems [Pnueli, FOCS 77].

- Model-checking approach:
  - Computer system is modelled as a graph/model $\mathcal{M}$.
  - Specification is a temporal logic formula φ.
  - Check whether $\mathcal{M}$ satisfies φ ($\mathcal{M} \models \varphi$).

- Automata-based approach
  (Gödel prize 2000) [Vardi & Wolper, IC 94]

- Early work on logic and automata. [Büchi, 62]

In this Course: Focus on Counter Systems

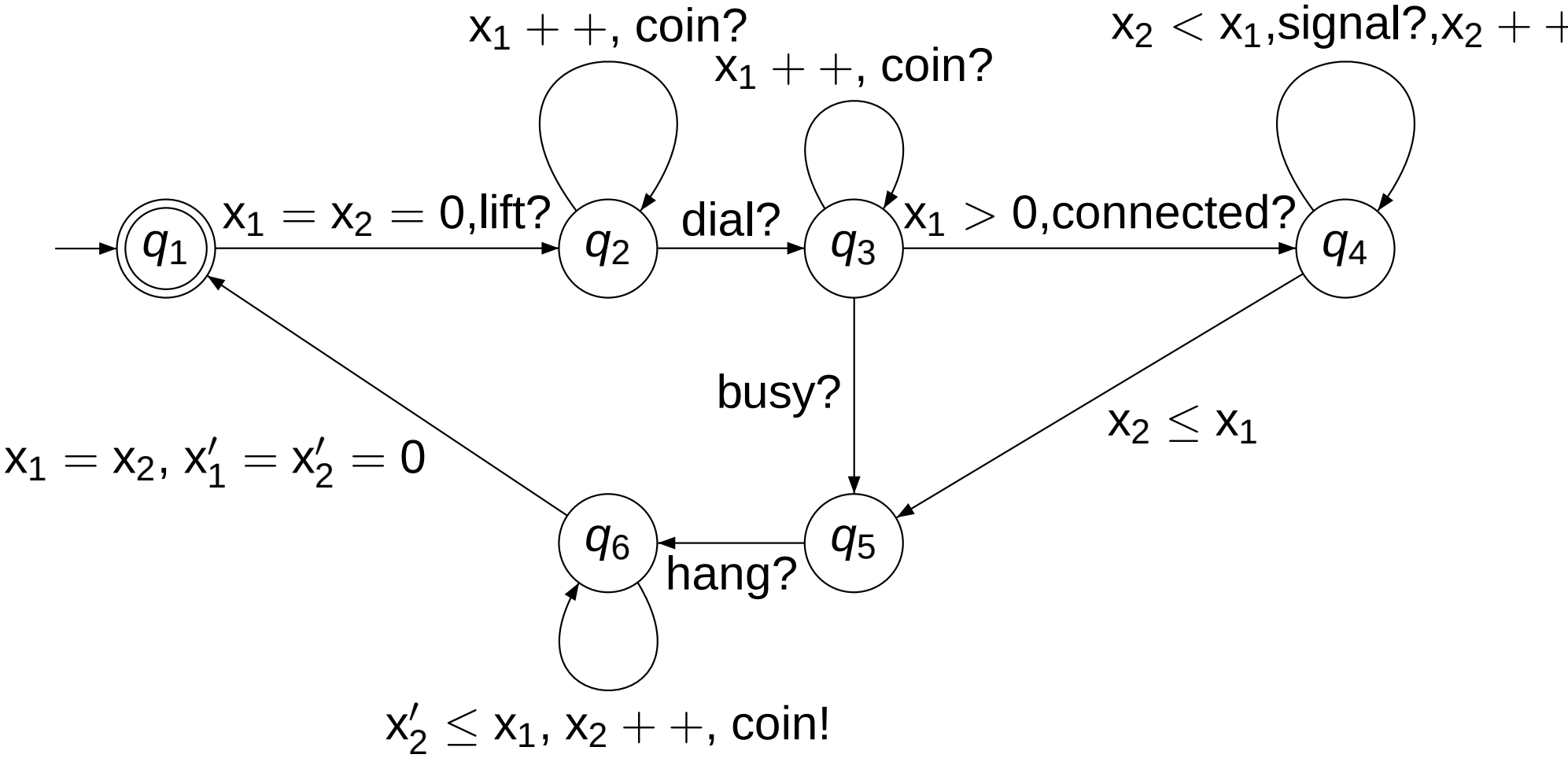
Ubiquity of counter systems

- Counter system: finite-state automaton with counters interpreted by nonnegative integers.
- Techniques for model-checking infinite-state systems are required for formal verification.
- Many applications:
  - Broadcast protocols, Petri nets, . . .
  - Programs with pointer variables. [Bouajjani et al., CAV'06]
  - Replicated finite-state programs. [Kaiser & Kroening & Wahl, CAV'10]
  - Relationships with data logics. [Bojańczyk et al., LICS'06]
  - . . .
- But, counter systems can simulate Turing machines.
- Checking safety or liveness properties for counter systems are undecidable problems.

Taming counter systems

- Design of subclasses with decidable reachability problems
  - Vector addition systems ($\approx$ Petri nets). [Kosaraju, STOC'82]
  - Flat relational counter systems. [Comon & Jurski, CAV'98]
  - Reversal-bounded counter automata. [Ibarra, JACM 78]
  - Flat affine counter systems. [Boigelot, PhD 98; Finkel & Leroux, FSTTCS'02]
  - **...**
- Decision procedures
  - Translation into Presburger arithmetic.
  - Direct analysis on runs. [Rackoff, TCS 78]
  - Approximating reachability sets. [Karp & Miller, JCSS 69]
  - Well-structured transition systems. [Finkel & Schnoebelen, TCS 01]
- Tools: FAST, LASH, TREX, ...

Toy Example: Pay Phone Controller



- x_1: number of coins which have been inserted.
- x_2: number of time units spent for communication.
- x_1' [resp. x_2'] is the next value of x_1 [resp. x_2].
- $x_1 + +$ is a shortcut for $x_1' = x_1 + 1 \wedge x_2' = x_2$.

How to read the figure

- q_1 is the initial state and the final state.
- x_1 and x_2 can only take nonnegative values.
- The controller interacts with the environment including the phone box. It can receive or send messages.
- Message ’coin?’: the controller receives the information that a coin has been inserted.
- Message ’coin!’: the controller sends the information that a coin has been released.

Underlying infinite transition system

- Configuration: description of the current state of the system.
- A configuration is a triple (q, n_1, n_2) where q is a control state and n_1 [resp. n_2] is the value of x_1 [resp. x_2].
- Because of the presence of messages, queues for messages should be added (omitted here).
- An execution is a (possibly infinite) sequence of configurations constrained by the system.
- Unbounded insertion of coins:

$$(q_1, 0, 0), (q_2, 0, 0), (q_2, 1, 0), (q_2, 2, 0), (q_2, 3, 0), \ldots$$

- This system is a finite and concise representation of an infinite labeled transition system.

Which properties hold true?

- Total communication time is never greater than the number of inserted coins:

$$\mathsf{A}\,\mathsf{G}\,\neg(x_2 > x_1).$$

- For all infinite executions, the number of coins is infinitely often equal to zero:

$$\mathsf{A}\,\mathsf{G}\,\mathsf{F}\,(x_1 = 0).$$

- There is an execution of the controller such that the total communication time is always equal to zero:

$$\mathsf{E}\,\mathsf{G}\,(x_2 = 0).$$

- Whenever the communication is over, eventually the system can reach the initial configuration:

$$\mathsf{A}\,\mathsf{G}\,(q_5 \Rightarrow \mathsf{F}q_1).$$

- Whenever the control state q_1 is reached, $x_1 = x_2 = 0$ and conversely:

$$\mathsf{A}\,\mathsf{G}(q_1 \Leftrightarrow (x_1 = 0 \wedge x_2 = 0)).$$

A Fundamental Model: Minsky Machines

Deterministic Minsky machines

- A counter stores a single natural number.
- A Minsky machine can be viewed as a finite-state machine with two counters.
- Operations on counters:
  - Check whether the counter is zero.
  - Increment the counter by one.
  - Decrement the counter by one if nonzero.

2-counter Minsky machines

- Set of n instructions.
- The lth instruction has one of the forms below ($i \in \{1, 2\}$, $l' \in \{1, \ldots, n\}$):

  I: $C_i := C_i + 1$; goto l'

  I: if $C_i = 0$ then goto l' else $C_i := C_i - 1$; goto l''.

- Configurations are elements of $\{1, \ldots, n\} \times \mathbb{N} \times \mathbb{N}$.
- Initial configuration: $(1, 0, 0)$.

Computations

- A computation is a sequence of configurations starting from the initial configuration and such that two successive configurations respect the instructions.

- The Minsky machine

  1: $C_1 := C_1 + 1$; goto 2
  2: $C_2 := C_2 + 1$; goto 1

  has unique computation

$$(1,0,0) \to (2,1,0) \to (1,1,1) \to (2,2,1) \to (1,2,2) \to (2,3,2)\ldots$$

Halting problem

- Halting problem:

  input: a 2-counter Minsky machine M;
  question: is there a finite computation that ends with location equal to n?

  (n may also be a special instruction that halts the machine)

- Theorem: The halting problem is undecidable.

  [Minsky, book 67]

- Minsky machines are Turing-complete (see next slide).

Turing machines

- Nondeterministic Turing machine $M = (Q, q_0, \Sigma, \delta, q_a)$:
  - Q: set of control states.
  - q_0: initial state; q_a: accepting state.
  - Σ: tape symbols (including a blank symbol or an end symbol).
  - Transition relation $\delta : Q \times \Sigma \to \mathcal{P}(Q \times \overbrace{\{-1, 0, 1\}}^{\text{moves}} \times \Sigma)$.
- We can assume that the Turing machine starts with an "empty" tape.
- The halting problem for Turing machines is undecidable [Turing, 1936].

Simulating a Turing machine (ideas only)

- A Turing machine can be simulated by two stacks (the tape is cut in half).
  - E.g., moving the head left or right is equivalent to popping a bit from one stack and pushing it onto the other

- A stack over a binary alphabet can be simulated by two counters. One counter contains the binary representation of the bits on the stack.
  - E.g., pushing a one is equivalent to doubling and adding 1, assuming that in the binary representation the least significant bit is on the top.

- Four counters can be simulated by two counters.
  - Counter values (a, b, c, d) encoded by value $2^a 3^b 5^c 7^d$.
  - E.g., checking the third counter is zero is equivalent to dividing by 5 and see what the remainder is. The second counter is auxiliary.

Non-deterministic Minsky machines

- Nondeterministic choice after incrementation and decrementation.

- Instructions are of the forms below:

  I: $C_i := C_i + 1$; goto I' or goto I''

  I: if $C_i = 0$ then goto I' else $C_i := C_i - 1$; goto I''_0 or goto I''_1.

- Recurrence problem:

  input: a NDM Minsky machine M;

  question: is there an infinite computation with instruction 1 occurring infinitely often?

- The recurrence problem is Σ^1_1-complete, i.e. highly undecidable. [Alur & Henzinger, JACM 94]

Minsky machines: an assembly language ?

- Minsky machines have a strong computational power.
- But, it is unlikely that one may wish to solve decision problems by programming Minsky machines.
- Problems on Minsky machines are easily undecidable.
- Counter systems will allow more flexibility and admit a richer set of instructions.
- . . . but, first we need to present Presburger arithmetic.

Presburger Arithmetic

A fundamental decidable theory

- First-order theory of $(\mathbb{N}, +)$ introduced by Mojcesz Presburger (1929).
- Instrumental to constraint counter values in counter systems.
- Formulae are viewed as symbolic representations for (infinite) sets of tuples of natural numbers.
- A first-order theory with many interesting properties:
  - Decidability (by contrast to first-order theory of $(\mathbb{N}, +, \times)$).
  - Sets definable in Presburger arithmetic are precisely semilinear sets (see next slides).
- Formalism also used to express constraints on graphs, on number of events, etc.

  See e.g., [Seidl & Schwentick & Muscholl, chapter 07]

Presburger arithmetic [Presburger, 29]

- "First-order theory of $(\mathbb{N}, +)$" (no multiplication).
- Terms: $t ::= 0 \mid 1 \mid \mathrm{x} \mid t + t$.
- $2\mathrm{x} + 3$ is a shortcut for $\mathrm{x} + \mathrm{x} + 1 + 1 + 1$.
- Presburger formulae ($k \geq 2$)

$$\varphi ::= t \equiv_k t \mid t < t \mid \neg\varphi \mid \varphi \wedge \varphi \mid \exists \mathrm{x}\, \varphi \mid \forall \mathrm{x}\, \varphi$$

- Valuation $\mathbf{v} : \mathrm{VAR} \to \mathbb{N}$ + extension to all terms with

$$\mathbf{v}(0) = 0 \quad \mathbf{v}(1) = 1 \quad \mathbf{v}(t + t') = \mathbf{v}(t) + \mathbf{v}(t')$$

- Oddness: $\exists \mathrm{y}\ \mathrm{x} = \mathrm{y} + \mathrm{y} + 1$.
  (with "$t = t'$" $\stackrel{\text{def}}{=}$ "$\neg(t < t' \vee t' < t)$")

Semantics

- $\mathbf{v} \models t \equiv_k t' \overset{\text{def}}{\Leftrightarrow}$ there is $m \in \mathbb{Z}$ such that $km + \mathbf{v}(t) = \mathbf{v}(t')$,
- $\mathbf{v} \models t < t' \overset{\text{def}}{\Leftrightarrow} \mathbf{v}(t) < \mathbf{v}(t')$,
- $\mathbf{v} \models \neg\varphi \overset{\text{def}}{\Leftrightarrow} \mathbf{v} \not\models \varphi$,
- $\mathbf{v} \models \varphi \wedge \varphi' \overset{\text{def}}{\Leftrightarrow} \mathbf{v} \models \varphi$ and $\mathbf{v} \models \varphi'$,
- $\mathbf{v} \models \exists \mathrm{x}\, \varphi \overset{\text{def}}{\Leftrightarrow}$ there is $n \in \mathbb{N}$ such that $\mathbf{v}[\mathrm{x} \mapsto n] \models \varphi$ where $\mathbf{v}[\mathrm{x} \mapsto n]$ is equal to $\mathbf{v}$ except that x is mapped to n,
- $\mathbf{v} \models \forall \mathrm{x}\, \varphi \overset{\text{def}}{\Leftrightarrow}$ for every $n \in \mathbb{N}$, we have $\mathbf{v}[\mathrm{x} \mapsto n] \models \varphi$.

$t \equiv_k t'$ is equivalent to $\exists\, \mathrm{x}\, (t = k\mathrm{x} + t') \vee (t' = k\mathrm{x} + t)$.

Defining sets of tuples

- Formula $\varphi(x_1, \ldots, x_n)$ with n free variables:

$$\mathrm{REL}(\varphi(x_1, \ldots, x_n)) \stackrel{\text{def}}{=} \{(\mathbf{v}(x_1), \ldots, \mathbf{v}(x_n)) \in \mathbb{N}^n : \mathbf{v} \models \varphi\}.$$

- φ is satisfiable $\stackrel{\text{def}}{\Leftrightarrow}$ there is $\mathbf{v}$ such that $\mathbf{v} \models \varphi$.

- φ is valid $\stackrel{\text{def}}{\Leftrightarrow}$ for all $\mathbf{v}$, we have $\mathbf{v} \models \varphi$.

- If φ has no free variable, then satisfiability is equivalent to validity.

- $\varphi(x_1, \ldots, x_n)$ is valid iff $\forall x_1, \ldots, x_n \; \varphi(x_1, \ldots, x_n)$ is satisfiable/valid.

Decidability and quantifier elimination

- Theorem: The satisfiability problem for Presburger arithmetic is decidable. [Presburger, 29]

- Every Presburger formula is **effectively equivalent** to a Presburger formula without first-order quantification. [Presburger, 29]

  (periodicity atomic formulae are needed here)

- Satisfiability problem for quantifier-free formulae is NP-complete. [Papadimitriou, JACM 81]

  See also [Borosh & Treybig, AMS 76]

- About other first-order theories
  - Skolem arithmetic $(\mathbb{N}, 0, 1, \times)$ is decidable.
  - $(\mathbb{Z}, 0, 1, <, +)$ is decidable.
  - $(\mathbb{N}, 0, 1, \times, +)$ is undecidable.

Semilinear sets

- A linear set X is defined by a basis $\vec{b} \in \mathbb{N}^k$ and a finite set of periods $\{\vec{p}_1, \ldots, \vec{p}_m\}$:

$$X = \{\vec{b} + \sum_{i=1}^{i=m} n_i \vec{p}_i : n_1, \ldots, n_m \in \mathbb{N}\}$$

- A semilinear set is a finite union of linear sets.

- A linear set:

$$\left\{ \begin{pmatrix} 3 \\ 4 \end{pmatrix} + i \times \begin{pmatrix} 2 \\ 5 \end{pmatrix} + j \times \begin{pmatrix} 4 \\ 7 \end{pmatrix} : i, j \in \mathbb{N} \right\}$$

- Subsets of $\mathbb{N}$ that are not semilinear:
  - $\{2^i : i \in \mathbb{N}\}$.
  - $\{i^2 : i \in \mathbb{N}\}$.

The fundamental characterization
[Ginsburg & Spanier, PJM 66]

- For every Presburger formula φ with $n \geq 1$ free variables, $\mathrm{REL}(\varphi)$ is a semilinear subset of $\mathbb{N}^n$.
- For every semilinear set $X \subseteq \mathbb{N}^n$, there is φ such that $X = \mathrm{REL}(\varphi)$.
- The class of semilinear sets are effectively closed under union, intersection, complementation and projection.
- For instance, ($X_1 = \mathrm{REL}(\varphi_1)$ and $X_2 = \mathrm{REL}(\varphi_2)$) imply $X_1 \cap X_2 = \mathrm{REL}(\varphi_1 \wedge \varphi_2)$
- Presburger formula for

$$\left\{ \begin{pmatrix} 3 \\ 4 \end{pmatrix} + i \times \begin{pmatrix} 2 \\ 5 \end{pmatrix} + j \times \begin{pmatrix} 4 \\ 7 \end{pmatrix} : i, j \in \mathbb{N} \right\}$$

$$\exists\, I, J\ (\mathrm{x}_1 = 3 + 2I + 4J \wedge \mathrm{x}_2 = 4 + 5I + 7J)$$

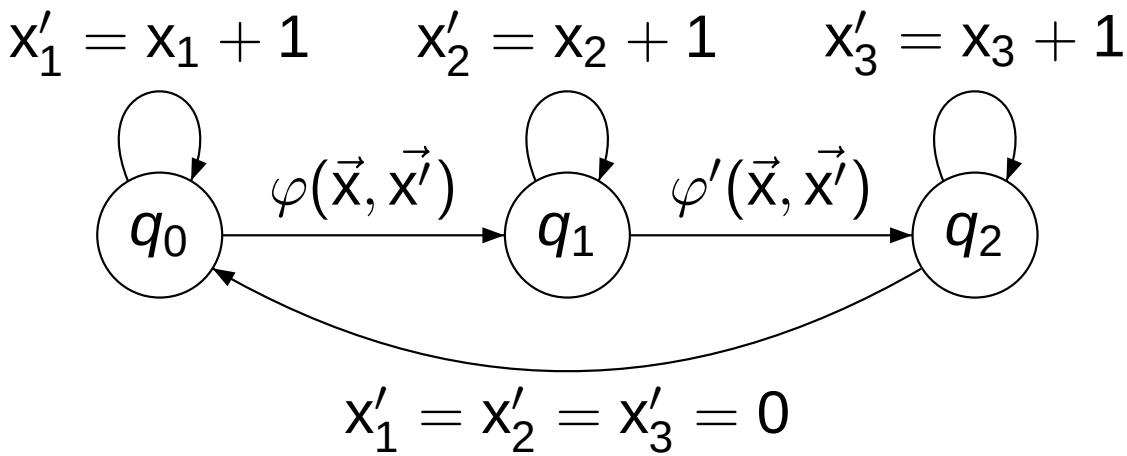
Parikh image

- $\Sigma = \{a_1, \ldots, a_k\}$ with ordering $a_1 < \cdots < a_k$.
- Parikh image of $u \in \Sigma^*$: $\begin{pmatrix} n_1 \\ n_2 \\ \vdots \\ n_k \end{pmatrix} \in \mathbb{N}^k$ where each n_j is the number of occurrences of a_j in u.
- Parikh image of $a\ b\ a\ a\ b$ is $\begin{pmatrix} 3 \\ 2 \end{pmatrix}$.
- Definition for Parikh image extends to languages.
- The Parikh image of any context-free language is semilinear. [Parikh, JACM 66]
- Effective computation from pushdown automata.

Counter Systems

Counter systems

- Counter system = finite-state automaton + counters governed by Presburger formulae.

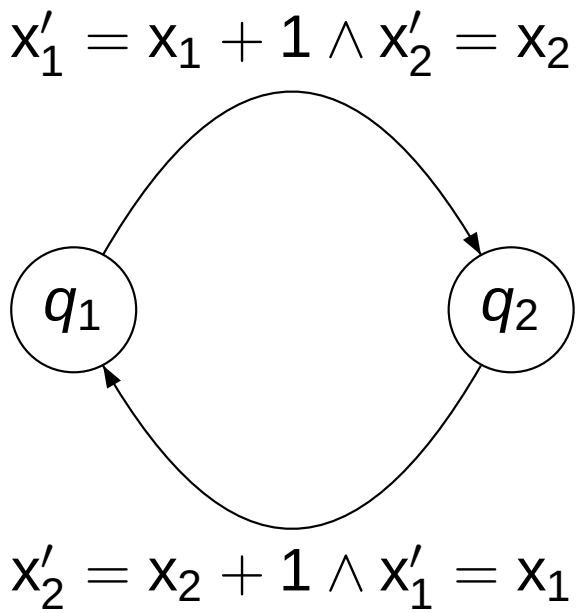


- Labels on transitions are Presburger formulae with
  - $\vec{x} = x_1, x_2, x_3$ (current values).
  - $\vec{x'} = x'_1, x'_2, x'_3$ (next values).

A simple counter system

1: $C_1 := C_1 + 1$; goto 2

2: $C_2 := C_2 + 1$; goto 1



A formal definition

- Counter system $\mathcal{S} = (Q, n, \delta)$ of dimension n:
  - Q is a nonempty finite set of control states.
  - $n \geq 1$ is the dimension.
  - δ is the transition relation: finite set of transitions of the form

$$t = (q, \varphi, q')$$

    where $q, q' \in Q$ and φ is a Presburger formula with free variables $x_1, \ldots, x_n, x'_1, \ldots, x'_n$.

- Prime variables are intended to be interpreted as the next values of the unprimed variables.

Interpretation: transition system

- Configuration $(q, \vec{y}) \in Q \times \mathbb{N}^n$.
- Let us define the valuation $\mathbf{v}_{\vec{y},\vec{y'}}$: for $i \in [1, n]$,
  1. $\mathbf{v}_{\vec{y},\vec{y'}}(\mathrm{x}_i) \stackrel{\text{def}}{=} \vec{y}(i)$,
  2. $\mathbf{v}_{\vec{y},\vec{y'}}(\mathrm{x}'_i) \stackrel{\text{def}}{=} \vec{y'}(i)$.
- Given $t = q \xrightarrow{\varphi} q'$, $(q, \vec{y}) \xrightarrow{t} (q', \vec{y'}) \stackrel{\text{def}}{\Leftrightarrow} \mathbf{v}_{\vec{y},\vec{y'}} \models \varphi$.
- Transition system $T(\mathcal{S}) = (S, \rightarrow)$
  - $S = Q \times \mathbb{N}^n$,
  - $(q, \vec{y}) \rightarrow (q', \vec{y'}) \stackrel{\text{def}}{\Leftrightarrow} \exists\ t \in \delta$ s.t. $(q, \vec{y}) \xrightarrow{t} (q', \vec{y'})$.
  - Reflexive and transitive closure $\xrightarrow{*}$.
- Runs as nonempty (possibly infinite) sequences

$$\rho = (q_0, \vec{y_0}) \rightarrow (q_1, \vec{y_1}) \cdots (q_k, \vec{y_k}) \cdots$$

Reachability problems

- REACHABILITY PROBLEM:
  Input: counter system $\mathcal{S}$, $(q, \vec{x})$ and $(q', \vec{x'})$.
  Question: is there a finite run with initial configuration $(q, \vec{x})$ and final configuration $(q', \vec{x'})$?
  (in symbols $(q, \vec{x}) \xrightarrow{*} (q', \vec{x'})$?)

- CONTROL STATE REACHABILITY PROBLEM:
  Input: counter system $\mathcal{S}$, $(q, \vec{x})$ and q'.
  Question: is there a finite run with initial configuration $(q, \vec{x})$ and whose final configuration has control state q'?
  $(\exists \vec{x'}\ (q, \vec{x}) \xrightarrow{*} (q', \vec{x'})$?)

- CONTROL STATE REPEATED REACHABILITY PROBLEM:
  Input: counter system $\mathcal{S}$, $(q, \vec{x})$ and q_f.
  Question: is there an infinite run with initial configuration $(q, \vec{x})$ such that the control state q_f is repeated infinitely often?

Variant problems

- COVERING PROBLEM:

  Input: counter system $\mathcal{S}$, $(q, \vec{x})$ and $(q', \vec{x'})$.

  Question: is there a finite run with initial configuration $(q, \vec{x})$ and whose final configuration is $(q', \vec{x''})$ with $\vec{x'} \preceq \vec{x''}$?

  (control state reachability is an instance with $\vec{x'} = \vec{0}$)

- BOUNDEDNESS PROBLEM:

  Input: counter system $\mathcal{S}$ and $(q, \vec{x})$.

  Question: is the set $\{(q', \vec{x'}) \in Q \times \mathbb{N}^n : (q, \vec{x}) \xrightarrow{*} (q', \vec{x'})\}$ finite?

- TERMINATION PROBLEM:

  Input: counter system $\mathcal{S}$ and $(q, \vec{x})$.

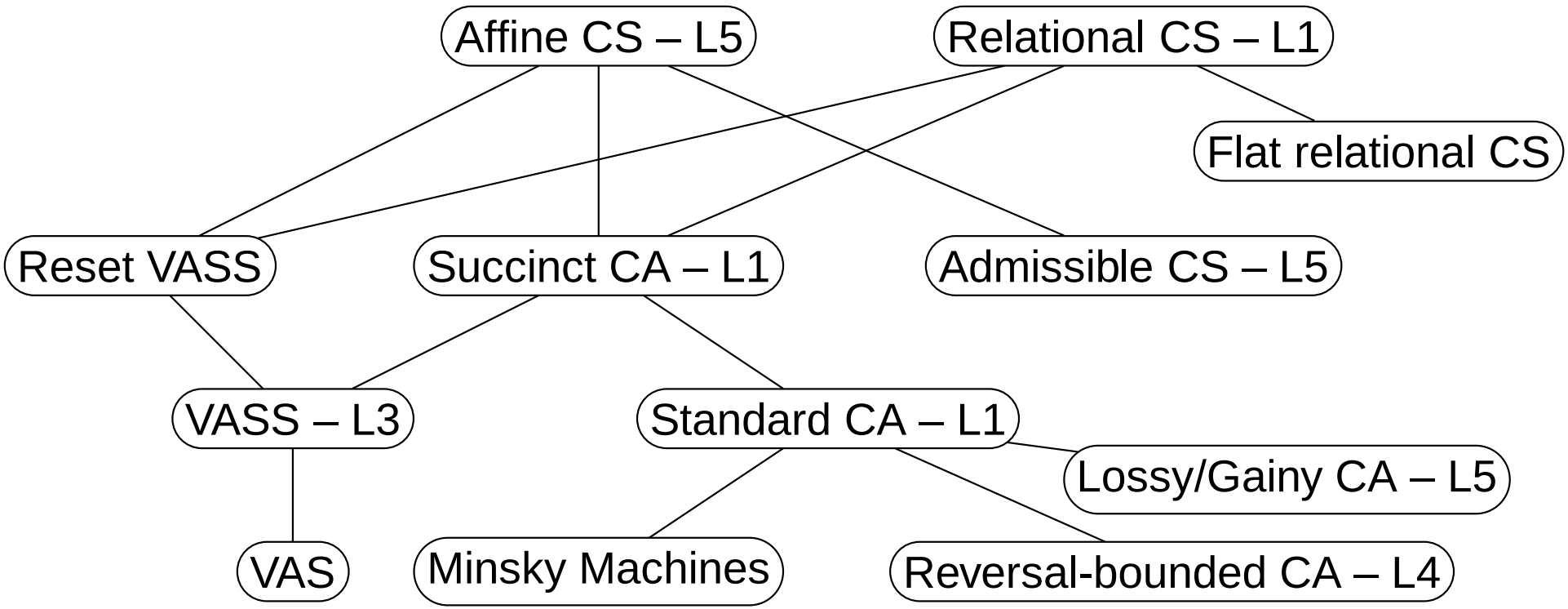
  Question: is there an infinite run with initial configuration $(q, \vec{x})$?

  Does termination implies boundedness?

What's next? . . . subclasses

- How to obtain subclasses:
  - restriction on syntactic ressources (number of counters, Presburger formulae etc.)
  - restriction on the control graph (e.g. flatness),
  - semantical restrictions (reversal-boundedness, etc.)
- Syntactic presentation of counter systems may be simplified (e.g., avoiding the use of Presburger formulae).

Classes of counter systems



Relational Counter Automata

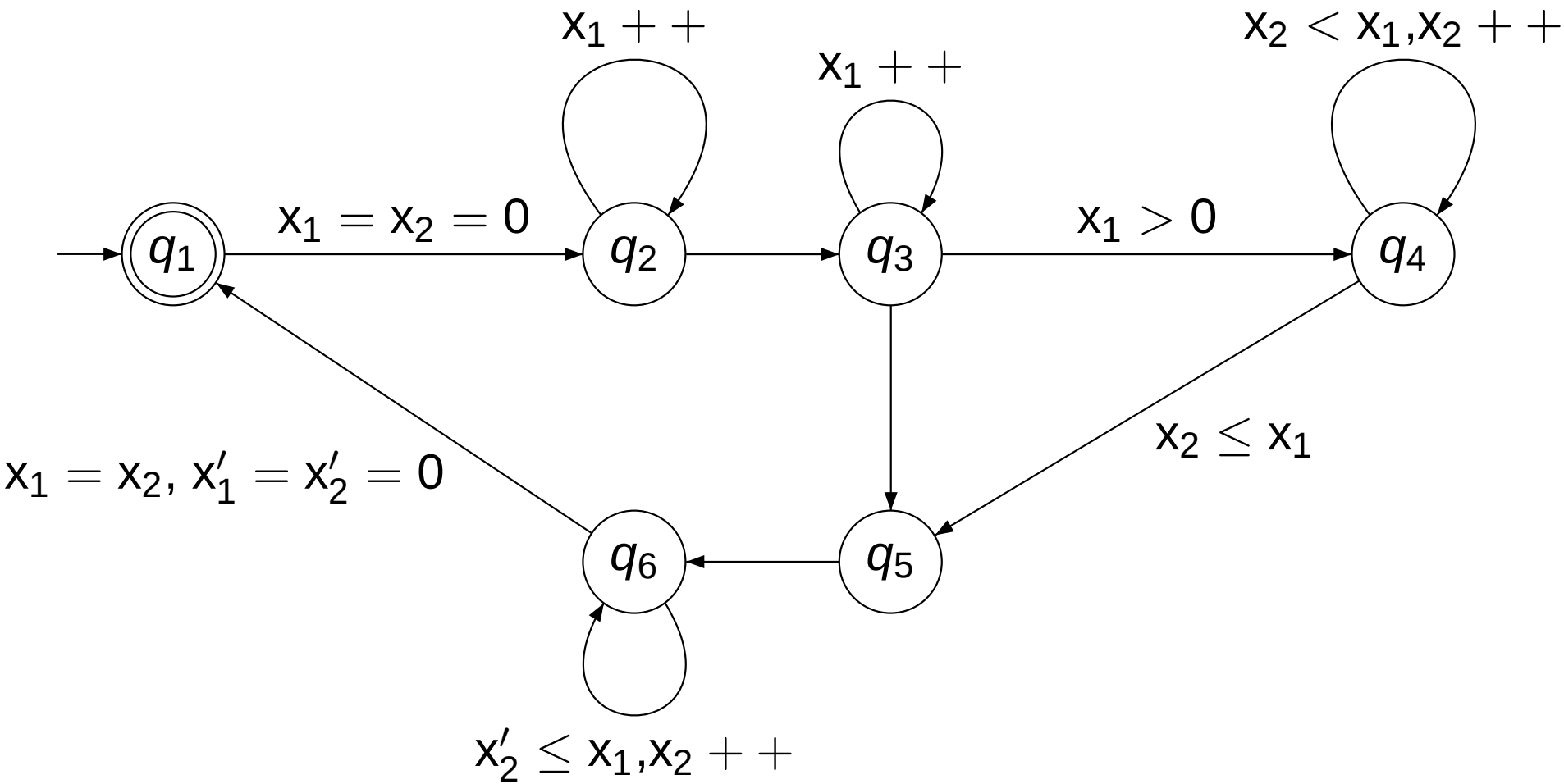
Nondeterministic update functions

- Relational counter system $\mathcal{S} = (Q, n, \delta)$: counter system such that for $q \xrightarrow{\varphi} q' \in \delta$, φ is a conjunction of atomic formulae of the form
  1. either $x \sim y + c$,
  2. or $x \sim c$,

  where $x, y \in \{x_1, \ldots, x_n, x'_1, \ldots, x'_n\}$, $c \in \mathbb{Z}$ and $\sim \in \{\geq, \leq, =, >, <\}$.

- Example ($n = 2$): $\varphi = (x_1 + 1 < x'_1) \wedge (x_2 - 3 = x'_2)$.

Phone controller is back !



Closure by composition [Comon & Jurski, CAV 98]

- $q \xrightarrow{x_1' = x_1 + 1} q'$ followed by $q' \xrightarrow{x_1' > x_1} q''$ is equivalent to

$$q \xrightarrow{x_1' \geq x_1 + 2} q''$$

- $q \xrightarrow{x_1' = x_2' = x_1} q'$ followed by $q' \xrightarrow{x_1' > x_1 \wedge x_2' > x_2} q''$ is equivalent to

$$q \xrightarrow{x_1' > x_1 \wedge x_2' > x_1} q''$$

- Generalization can be done as stated below.

- Lemma: Let $\mathcal{S}$ be a relational counter system.
  Given $t_1 = q \xrightarrow{\varphi_1} q'$ and $t_2 = q' \xrightarrow{\varphi_2} q''$, there is φ such that for all $\vec{x}$, $\vec{x'}$ and $\vec{x''}$ in $\mathbb{N}^n$, we have $(q, \vec{x}) \xrightarrow{t_1} (q', \vec{x'}) \xrightarrow{t_2} (q'', \vec{x''})$ iff $(q, \vec{x}) \xrightarrow{t} (q'', \vec{x''})$ with $t = q \xrightarrow{\varphi} q''$.

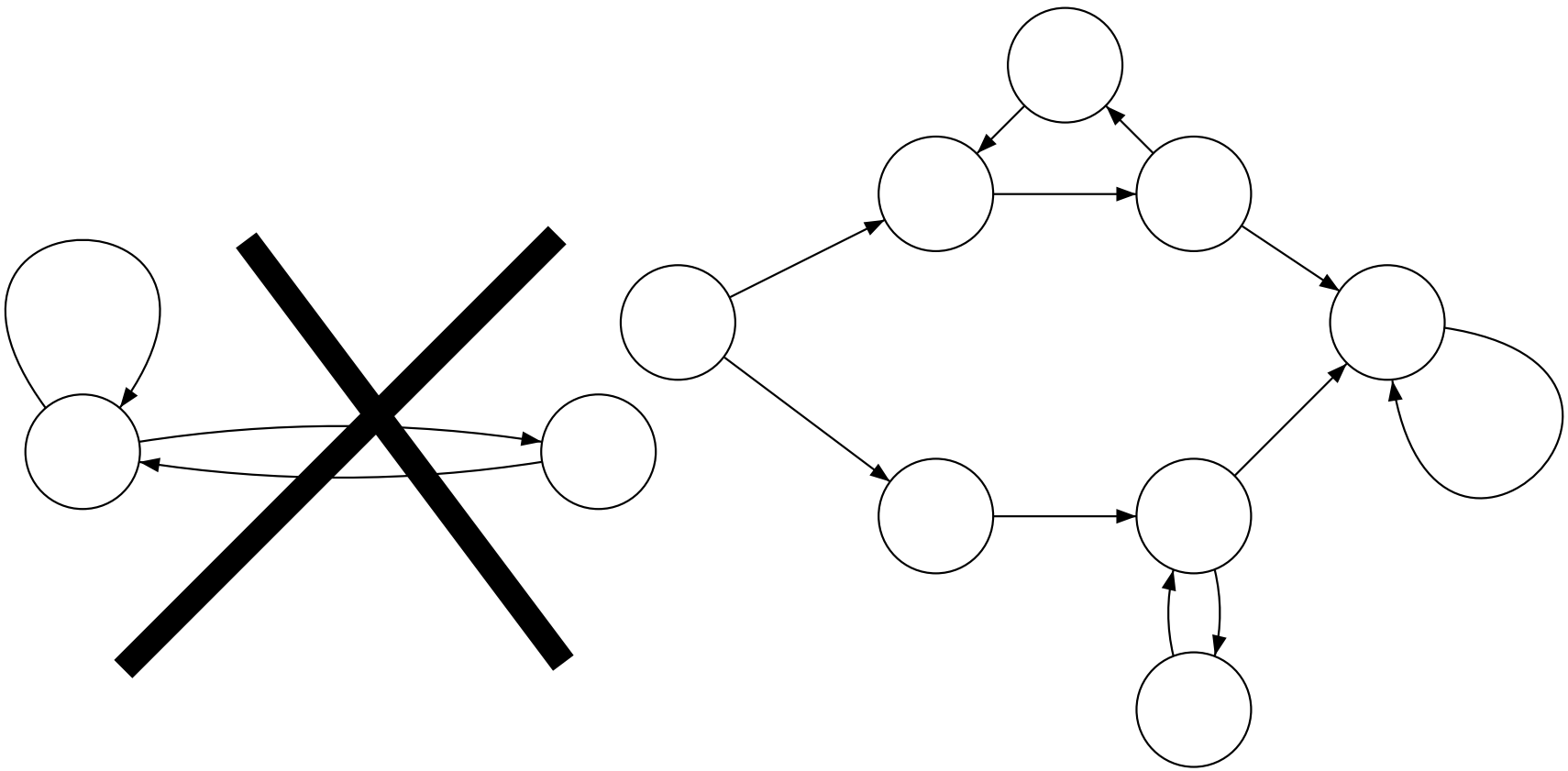
Closure by iteration in PrA

- With unique transition $t = q \xrightarrow{x_1' = x_1 + 1} q$, we have $(q, K) \xrightarrow{*} (q, K')$ iff $K' \geq K$.
- Finite iteration of t is $q \xrightarrow{x_1' \geq x_1 + 1} q$.
- With transition $t = q \xrightarrow{x_1' = x_1 + 2} q$, we have $(q, K) \xrightarrow{*} (q, K')$ iff there is $k \in \mathbb{N}$ such that $K' = K + 2k$.
- $(q, K) \xrightarrow{*} (q, K')$ iff $\mathbf{v}_{K,K'} \models \exists\, y\ x_1' = x_1 + 2 \times y$.
- Theorem: Let $\mathcal{S}$ be a relational counter system made of a unique transition $q \xrightarrow{\varphi} q$. One can effectively compute a **Presburger formula** φ' with free variables $x_1, \ldots, x_n, x_1', \ldots, x_n'$ s.t.

  for all $\vec{x}$, $\vec{x'}$ in $\mathbb{N}^n$, $(q, \vec{x}) \xrightarrow{*} (q, \vec{x'})$ iff $\mathbf{v}_{\vec{x},\vec{x'}} \models \varphi'$.

Flatness

A relational counter system is flat if every control state belongs to at most one simple cycle. Moreover, there is at most one transition between two control states.



Reachability relation is Presburger-definable
[Comon & Jurski, CAV 98]

- Theorem Let $\mathcal{S}$ be a **flat** relational counter system and $q, q' \in Q$. One can effectively compute a Presburger formula φ s.t. for every $\mathbf{v}$, we have
  $\mathbf{v} \models \varphi$ iff $(q, (\mathbf{v}(x_1), \ldots, \mathbf{v}(x_n))) \xrightarrow{*} (q', (\mathbf{v}(x'_1), \ldots, \mathbf{v}(x'_n)))$.
- The reachability problem for flat relational counter systems is decidable.
  - Consider instance $\mathcal{S}$, $(q, \vec{y})$ and $(q', \vec{y'})$.
  - Compute the Presburger formula φ as above.
  - Check satisfiability of the formula below:

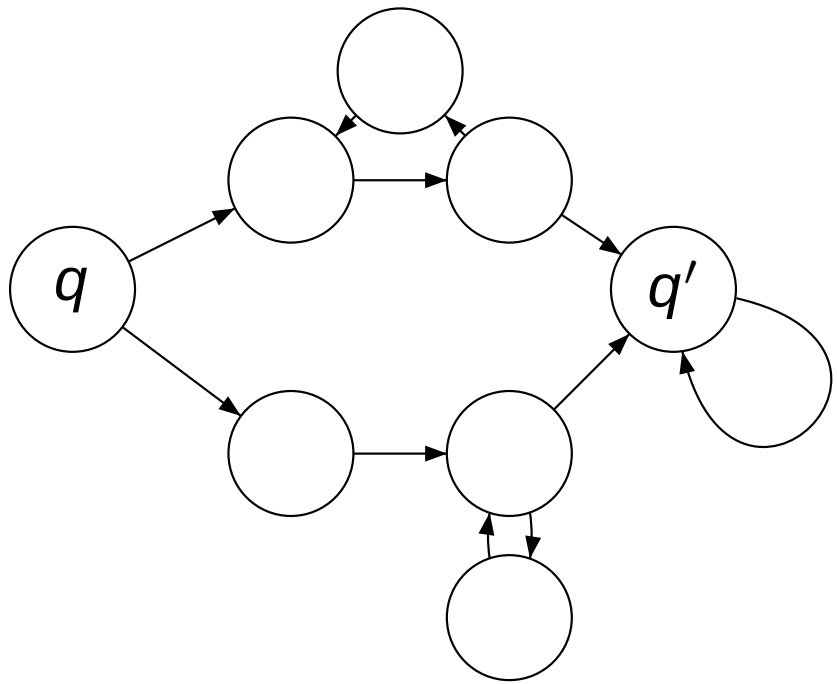
$$\left(\bigwedge_{i=1}^{i=n} (x_i = \vec{y}(i) \wedge x'_i = \vec{y'}(i))\right) \wedge \varphi$$

    assuming free variables in φ are $x_1, \ldots, x_n, x'_1, \ldots, x'_n$.

Proof sketch for the theorem

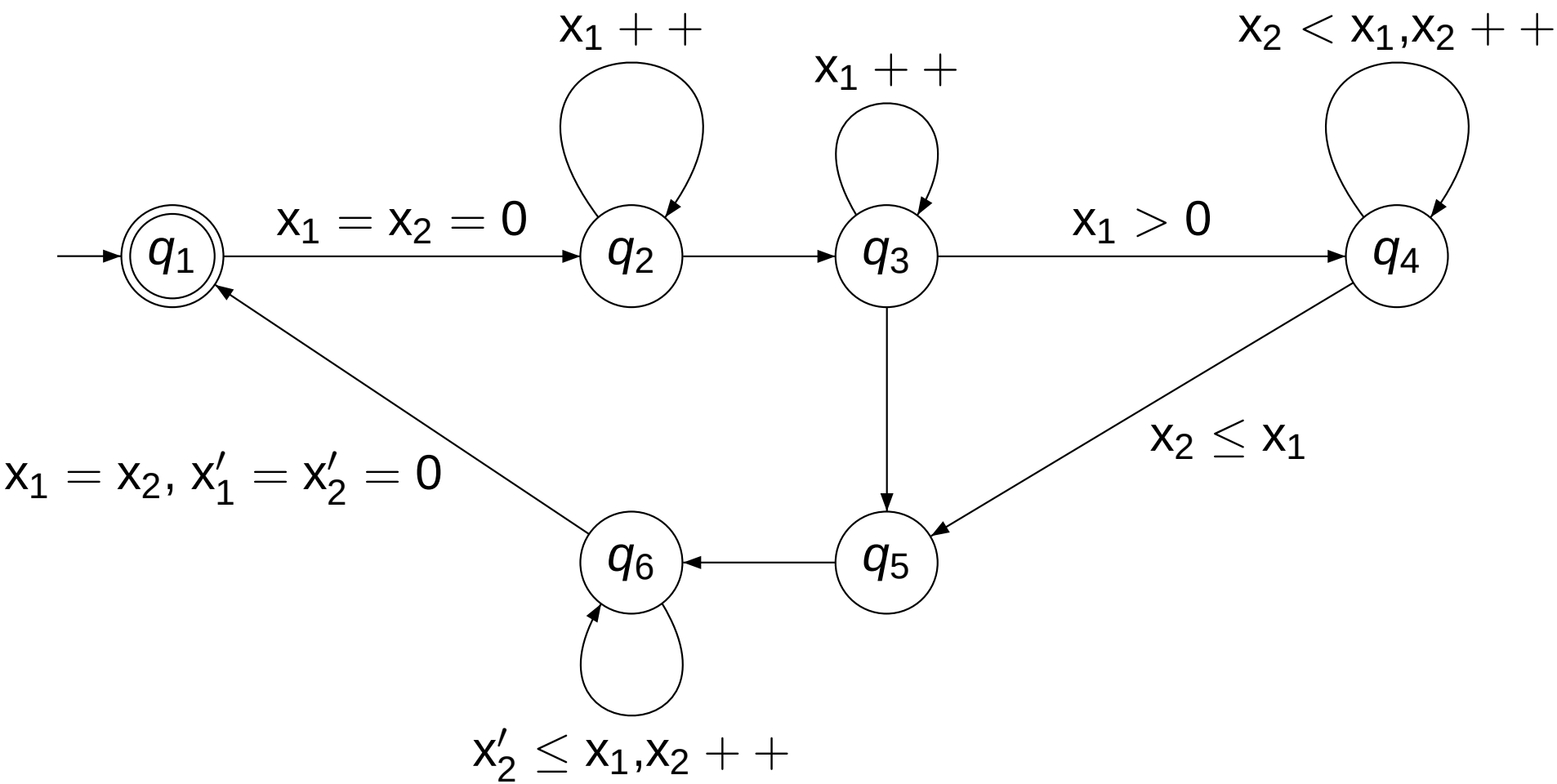
- For each cycle $q_1 \xrightarrow{\varphi_1} q_2 \xrightarrow{\varphi_2} \ldots \xrightarrow{\varphi_N} q_N$ ($q_1 = q_N$) compute the equivalent transition (q_1, φ, q_1).

- For q, q', enumerate the run schemata between q and q'



- Compute the formula for reachability relation by composition.

Is $\{\vec{x} \in \mathbb{N}^2 : (q_1, \vec{0}) \xrightarrow{*} (q_i, \vec{x}), i \in [1, 6]\}$ semilinear?



Counter Automata

Standard counter automata

- Standard counter automaton (Q, n, δ): transitions are of the form either $q \xrightarrow{\mathrm{inc}(i)} q'$ or $q \xrightarrow{\mathrm{dec}(i)} q'$ or $q \xrightarrow{\mathrm{zero}(i)} q'$ where
  - $\mathrm{inc}(i)$ is a shortcut for $(\mathsf{x}_i' = \mathsf{x}_i + 1) \wedge (\bigwedge_{j \neq i} \mathsf{x}_j' = \mathsf{x}_j)$,
  - $\mathrm{dec}(i)$ is a shortcut for $(\mathsf{x}_i' = \mathsf{x}_i - 1) \wedge (\bigwedge_{j \neq i} \mathsf{x}_j' = \mathsf{x}_j)$,
  - $\mathrm{zero}(i)$ is a shortcut for $(\mathsf{x}_i = 0) \wedge (\bigwedge_{j} \mathsf{x}_j' = \mathsf{x}_j)$.
- Minsky machines are standard counter automata.

Succinct counter automata

- Each transition either performs zero-tests on a subset of counters or updates counters by adding a vector in $\mathbb{Z}^n$.
- Succinct counter automaton (Q, n, δ): transitions of the form either $q \xrightarrow{\mathrm{inc}(\vec{b})} q'$ with $\vec{b} \in \mathbb{Z}^n$ or $q \xrightarrow{\mathrm{zero}(\vec{b'})} q'$ with $\vec{b'} \in \{0,1\}^n$ where
  - $\mathrm{inc}(\vec{b})$ is a shortcut for $\bigwedge_{i \in [1,n]} \mathsf{x}'_i = \mathsf{x}_i + \vec{b}(i)$,
  - $\mathrm{zero}(\vec{b'})$ is a shortcut for $\bigwedge_{i \in [1,n] \text{ s.t. } \vec{b'}(i)=1} \mathsf{x}_i = 0 \wedge \bigwedge_{i \in [1,n]} \mathsf{x}'_i = \mathsf{x}_i$
- Morally, standard counter automata and succinct counter automata are identical but there may be differences for complexity issues.

Vector Addition Systems with States (VASS)

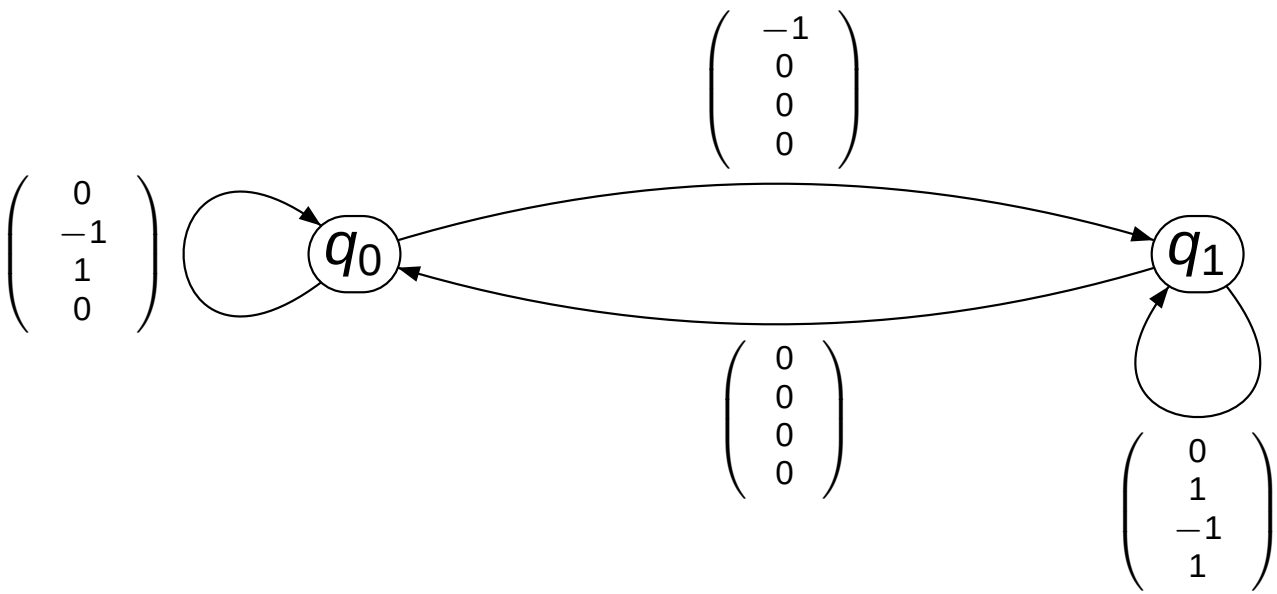
What is a VASS?

- VASS = finite-state automaton + translations of counters.
- VASS is a counter system with transitions of the form $q \xrightarrow{\vec{b}} q'$ with $\vec{b} \in \mathbb{Z}^n$, which is a shortcut for

$$\bigwedge_{i \in [1,n]} x_i' = x_i + \vec{b}(i)$$

- VAS = VASS with a unique control state.
- Petri nets, VAS and VASS are equivalent models.

Example



Can q_0, $\begin{pmatrix} 0 \\ 0 \\ 20 \\ 80 \end{pmatrix}$ be reached from q_0, $\begin{pmatrix} 4 \\ 20 \\ 0 \\ 0 \end{pmatrix}$?

Decidability/complexity issues

- Theorem: The reachability problem is decidable.
  [Mayr, STOC 81; Kosaraju, STOC 82]
  - No primitive recursive algorithm is known.
    (use of well quasi-orderings)
  - EXPSPACE-hardness [Lipton, TR 76].
- Theorem: The covering and boundedness problems for VASS are EXPSPACE-complete.
  [Lipton, TR 76; Rackoff, TCS 78]
  - Decidability shown in [Karp & Miller, JCSS 69].
  - EXPSPACE upper bound for path sublogic [Faouzi Atig & Habermehl, RP 09], correcting [Yen, IC 92].
- Checking equality between accessibility sets of two configurations is undecidable [Hack, TCS 76].

A few more remarks

- Control-state reachability is an instance of the covering problem.
- EXPSPACE-hardness holds true even with coefficients -1, 0 and 1 only.
- Boundedness and reachability problems are undecidable for VASS with resets.

  [Dufourd & Finkel & Schnoebelen, ICALP 98].
- Boundedness implies that the transition system from $(q, \vec{x})$ is equivalent to a finite-state automaton.

Conclusion

- Today's lecture:
  - Classes of counter systems and decision problems.
  - Presburger arithmetic.
- Tomorrow's lecture:
  - Standard LTL
  - Logic $\mathrm{LTL}^{\mathrm{CS}}(\mathrm{PrA})$ for counter systems
  - Presburger LTL
  - LTL with registers