Temporal Specification with Accumulative Values

Udi Boker

Joint work with
Krishnendu Chatterjee, Thomas Henzinger and Orna Kupferman

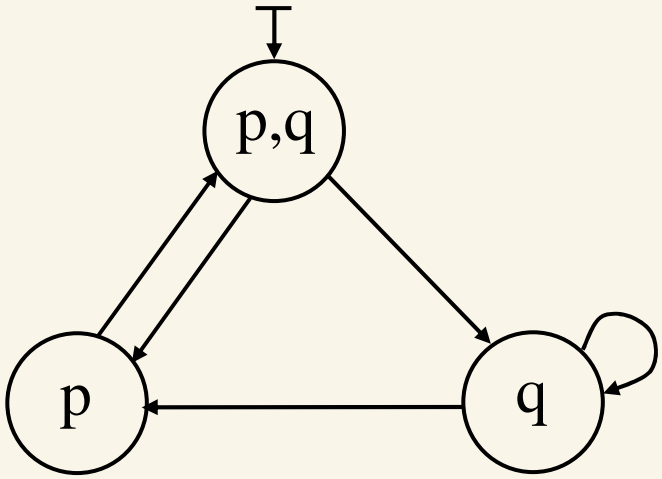
LICS 2011

The Current Paradigm

Model Checking (Bool-System, Bool-Spec)

Boolean Systems

E.g. Kripke structure:



Boolean Specifications

E.g. LTL formula: $G(p \rightarrow Fq)$

CTL formula: $EF(q \wedge AX(p))$

Good, But Not Enough

- Verifying finite-state systems and temporal specifications is very useful.
- Yet, it has a significant limitation of only handling Boolean systems and specifications.
- There is a growing demand for handling quantitative systems and specifications.

Partial Answer: Automata Specifications

- Currently, specific quantitative objectives are handled by means of acceptance conditions to weighted automata:
  - Energy
  - Mean payoff (Limit average)
  - Discounted sum
- The Cons:
  - Specific objectives.
  - No integration within a specification language.
  - Automata are less friendly as specifications.

Our Goal: Quantitative Specifications

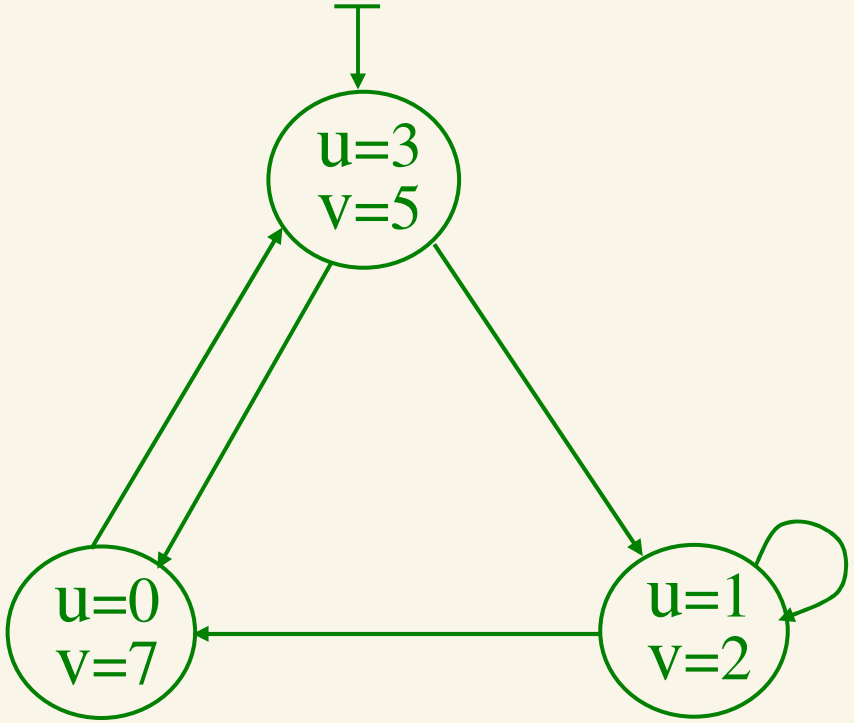
We want to enrich
temporal-logic specifications
with quantitative assertions.

E.g. "The total energy is always positive
and the average income is
eventually above 10,000 or …"

Model Checking ($\hat{Q}$uan-System, $\hat{Q}$uan-Spec)

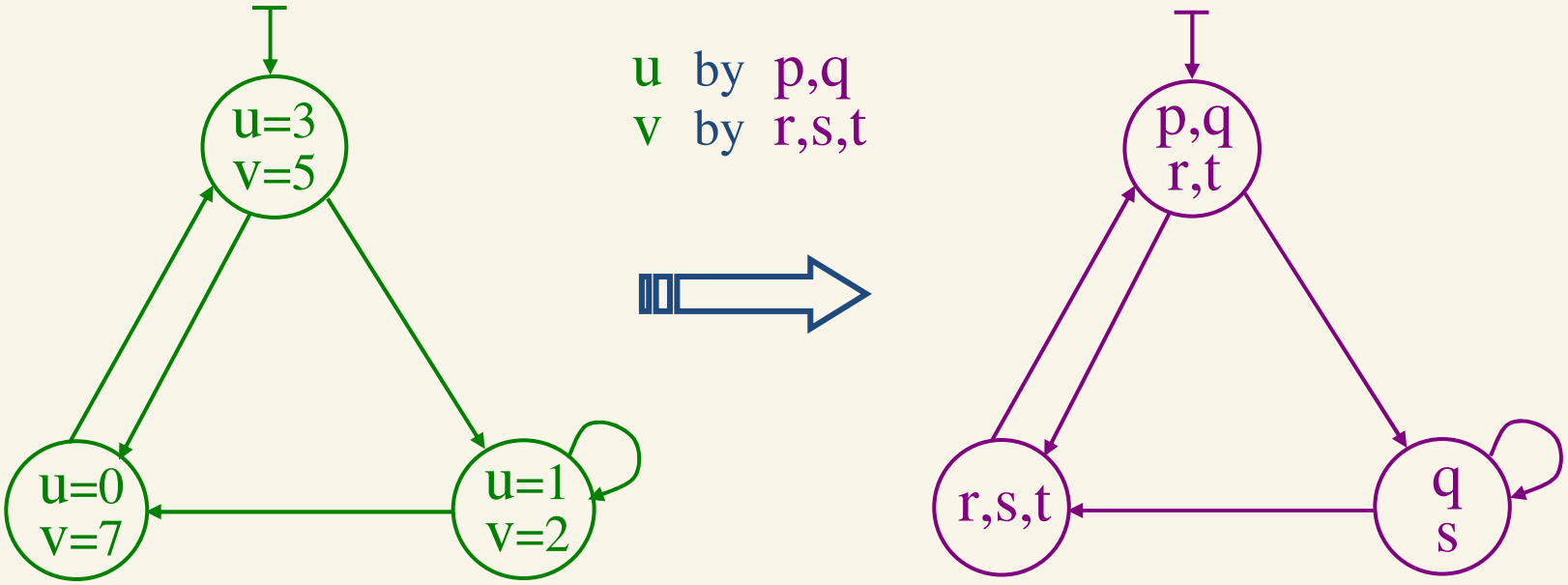
Quantitative Systems

- Having numeric variables instead of Boolean ones.



Quantitative/Boolean Systems

- Systems usually use rich variables including numbers.
- For Boolean specifications the numeric variables are represented by Boolean variables.



"Eventually $v<7$" by "Eventually $\neg r \vee \neg s \vee \neg t$"

Quantitative Kripke Structures

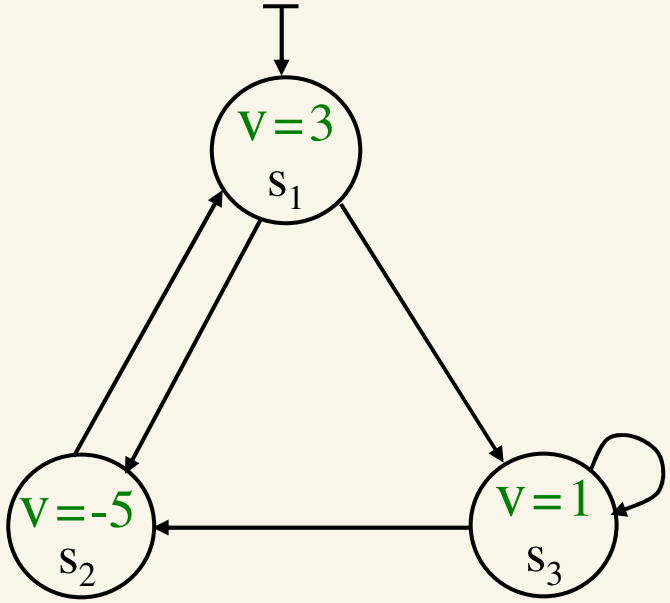
- Local numeric properties, as "$\nu<7$", are easily captured by the ω-regular paradigm.
- However, "The total sum of $\nu < 7$" is no longer ω-regular-definable.
  - It is based on an accumulative property.
  - It relates to an unbounded number of values.
  - It refers to the "numerical meaning" of ν.
- Quantitative-specification relates, in general, to quantitative Kripke structures.

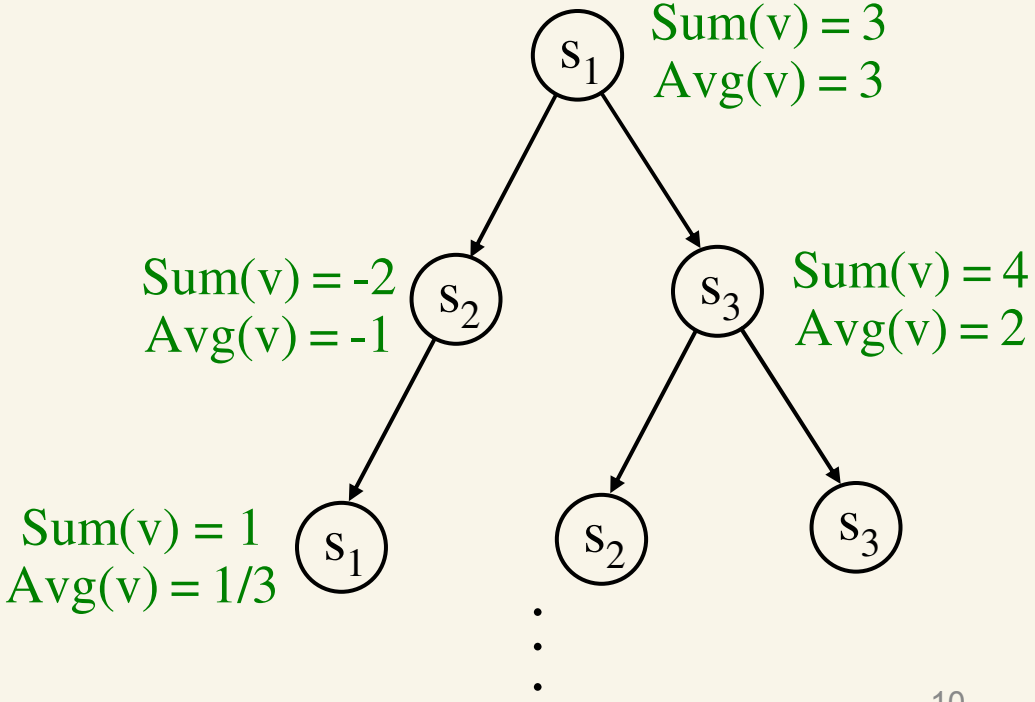
Accumulative Properties - Motivation

- Accumulation lies in the heart of quantitative objectives.
  - Summation (energy): $\sum_{i=1}^{n} x_i$
  - Average (mean payoff): $\frac{1}{n}\sum_{i=1}^{n} x_i$
  - Discounted-sum: $\sum_{i=1}^{n} \lambda^n x_i$
- Summarizes entire computation-prefix.
- Highly interesting for high-level specifications.

Sum/Avg Semantics

- We consider the computation tree of a quantitative Kripke structure.
- The value of Sum(v) at a tree node is the summation of v along the route to the node.
- The value of Avg(v) is Sum(v) divided by the route's length.





A Specific Goal

- Enrich linear/branching temporal logic with the following atomic assertions (c is a constant):
  - $\mathrm{Sum}(v) \geq c$; $\mathrm{Sum}(v) \geq \mathrm{Sum}(u)$
  - $\mathrm{Avg}(v) \geq c$; $\mathrm{Avg}(v) \geq \mathrm{Avg}(u)$
- Examples:
  - Linear: $G(p \rightarrow (q \vee \mathrm{Sum}(u)=5 \wedge \mathrm{Avg}(v)<3))$
  - Branching: $EF(\mathrm{Avg}(u)<\mathrm{Avg}(v) \wedge AG(p \rightarrow q))$

Avg ⇔ Sum

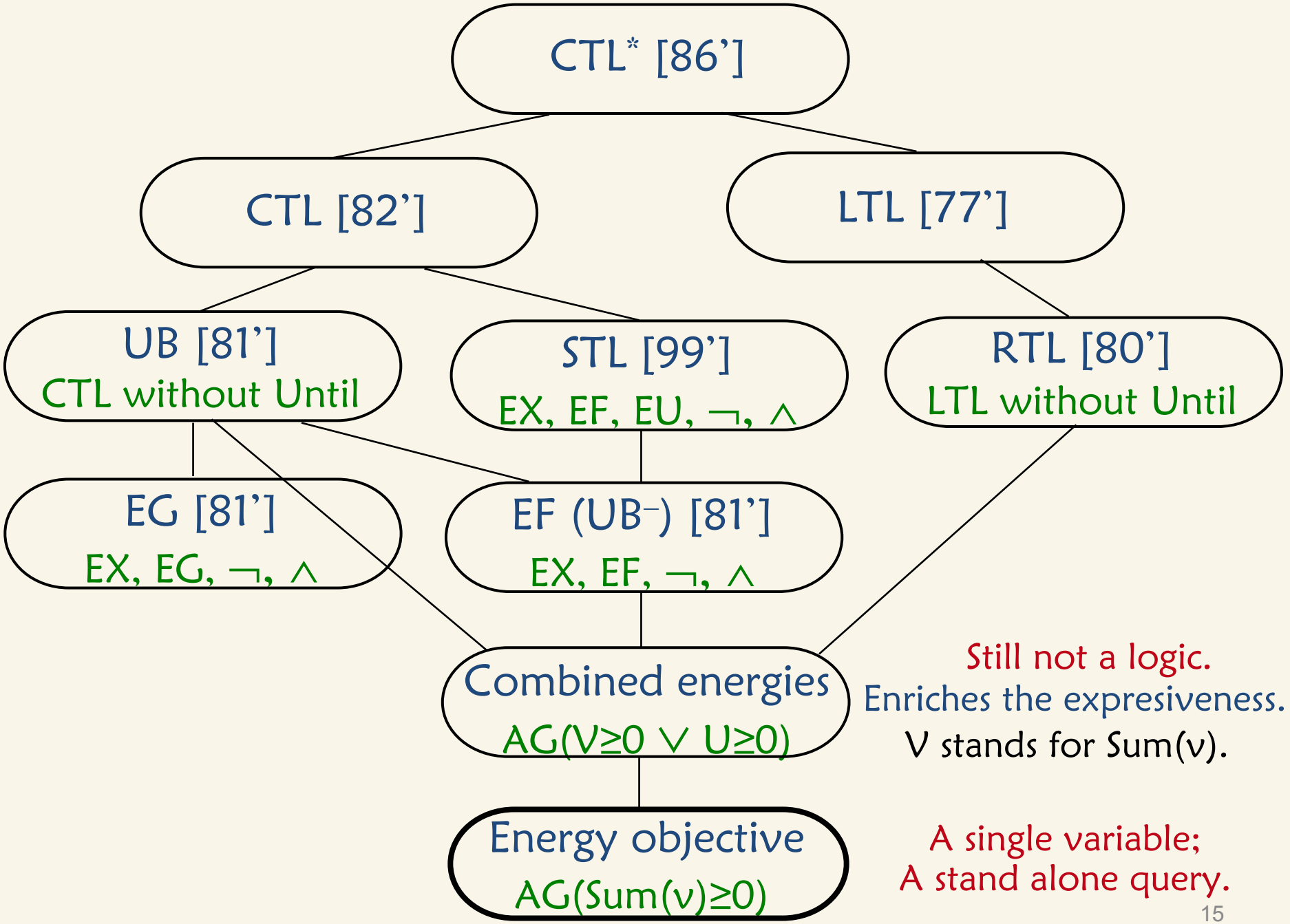
- We can express Avg propositions with Sum propositions and vice versa:
  - Avg(v) ≥ c iff Avg(v') ≥ 0
    for a new variable v' = v-c in all states.
  - Avg(v') ≥ 0 iff Sum(v') ≥ 0
- Comparing between accumulative variables is reduced to comparing against a constant:
  - Sum(u) ≥ Sum(v) iff Sum(d)≥0
    for a new variable d=u-v in all states.

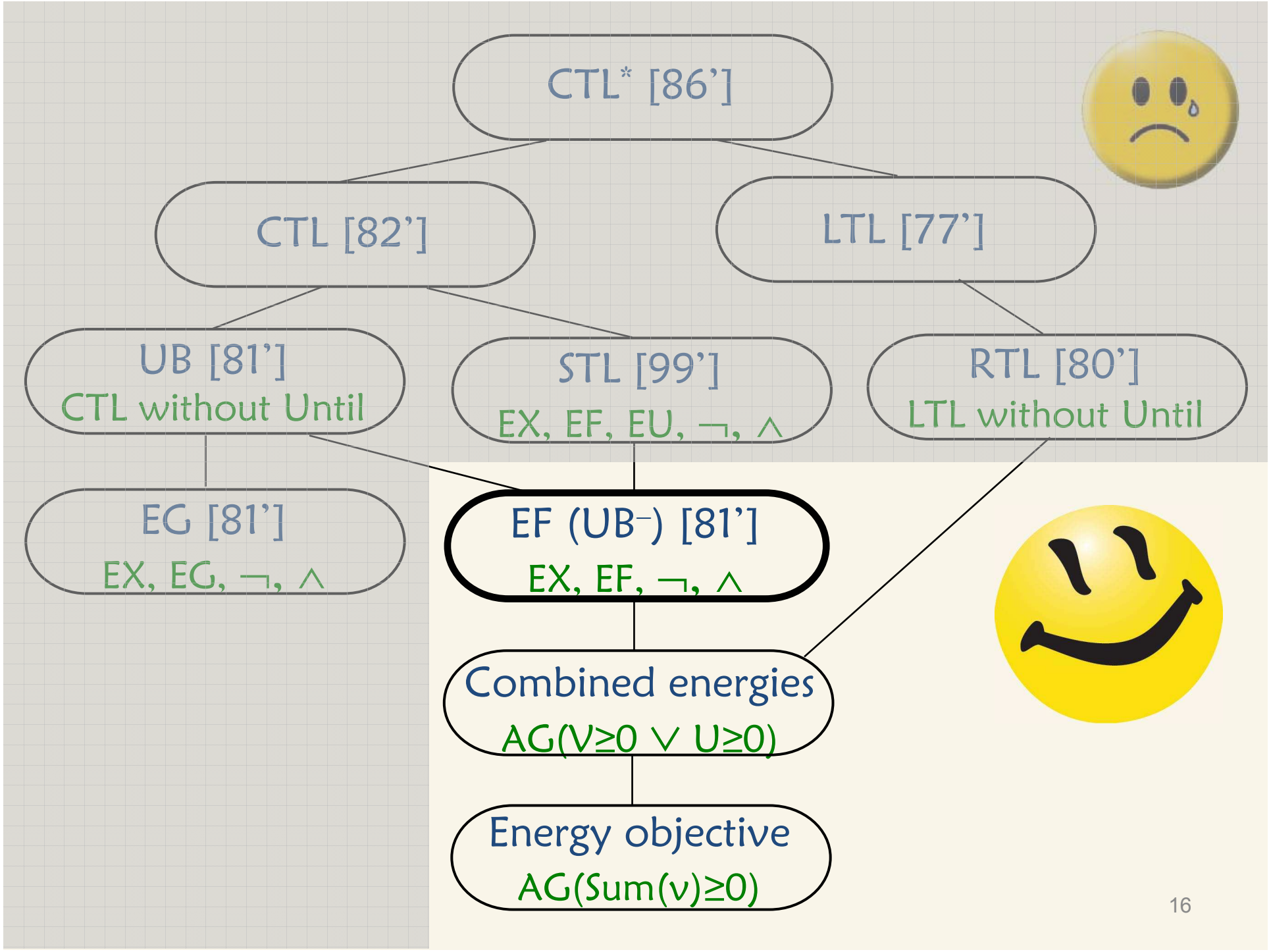
A Specific Question

- Which linear/branching temporal logic can be enriched with $Sum(v) \geq 0$, while allowing for a decidable model-checking procedure?
  - CTL^* ?
  - LTL ?
  - CTL ?
  - Any standard temporal logic ?
- The question may also refer to a Boolean Kripke structure, adding to the logic the atomic assertion $Avg(p) \geq ½$.

Some Hints (or not…)

- Kripke structures with accumulative sums take us from the comfort finite-system zone to the hazardous inifinite-system zone.
  - They are almost like counter machines, with the crucial difference of not branching by the sum-values.
  - They are almost like Petri-nets, with the difference that the latter must always have positive sums.
- Basic questions on multiple-counter machines are undecidable [Minsky 67].
- Model checking Petri-nets is undeciable with respect to all relevant temporal logics [Esparza 95].


CTL* [86']
CTL [82']
LTL [77']
UB [81']
CTL without Until
STL [99']
EX, EF, EU, ¬, ∧
RTL [80']
LTL without Until
EG [81']
EX, EG, ¬, ∧
EF (UB⁻) [81']
EX, EF, ¬, ∧
Combined energies
AG(V≥0 ∨ U≥0)
Energy objective
AG(Sum(v)≥0)
Still not a logic.
Enriches the expresiveness.
V stands for Sum(v).
A single variable;
A stand alone query.


CTL* [86']
CTL [82']
LTL [77']
UB [81']
CTL without Until
STL [99']
EX, EF, EU, ¬, ∧
RTL [80']
LTL without Until
EG [81']
EX, EG, ¬, ∧
EF (UB⁻) [81']
EX, EF, ¬, ∧
Combined energies
AG(V≥0 ∨ U≥0)
Energy objective
AG(Sum(v)≥0)

Maximal Decidable Temporal Logic

- The EF logic (EX, EF, $\neg$, $\wedge$) characterizes the temporal logic allowing for a decidable model checking of accumulative properties.

- A logic with any of the other standard temporal operators, EG, EU, ER or EW, is undecidable.

Controlled Accumulation

- One may wish to control the accumulation.
  - Taking the average value of some variable v only within "transactions".
  - Relating to the average *response time* between a "request" and a "grant".
- In general: $\dfrac{\text{Sum}\,(v \mid r_1)}{\text{Sum}\,(u \mid r_2)} \geq c$

  for numeric variables u and v, regular expressions r_1 and r_2, and constant c.
- This is decidable with the enriched EF logic!

Part I - Summary

- Specifications with accumulative propositions are decidable with the EF logic.
- It significantly enriches the currently handled energy objectives and controlled accumulation.
  - $AG(p \rightarrow (q \vee Sum(u)=5 \wedge Avg(v)<3))$
  - $EF(Avg(u)<Avg(v) \wedge AG(p \rightarrow q))$
  - Average response time ≤ 5
- All other temporal operators are undecidable.
  - In particular, EF does not capture the LimAvg (mean-payoff) objectives.

Agenda

Part I - Sum(v)≥c and Avg(v)≥c as atomic assertions.

- Proofs

Part II - A different approach, handling LimAvg.

- Proofs

Part II

Temporal logic with LimAvg

Mean Payoff / LimAvg

What is LimAvg(x) or mean-payoff?

- Intuitively, it is the long-run average of x in an infinite run.
- That is, the limit of Avg(x) along the run prefixes.

However, such a limit need not exist.

LimAvg Need Not Converge

The run r:

1 visit in q, 2 visits in q', 4 visits in q, 8 visits in q', …

LimInfAvg and LimSupAvg

- A possible approach is to use Inf and Sup [CDH08]:
  - $\mathrm{Avg}_n(x) = \frac{1}{n}\sum_{i=1}^{n} x_i$
  - $\mathrm{LimInfAvg}(x) = \liminf_{m\to\infty} \{\mathrm{Avg}_n(x) \mid n \geq m\}$
  - $\mathrm{LimSupAvg}(x) = \limsup_{m\to\infty} \{\mathrm{Avg}_n(x) \mid n \geq m\}$
- We shall write LimAvg when speaking of both LimInfAvg and LimSupAvg.

[CDH08] "Quantitative Languages" by Chatterjee, Doyen and Henzinger

LimAvg In LTL

- LimAvg is a path property.
- Hence, it fits well into an LTL formula.
- We would have liked to add $\text{LimAvg}(x) \geq c$ as an atomic assertion in LTL.
- Example:
  - $F(p) \rightarrow \text{LimAvg}(x) = 5 \ \vee \ G(q) \wedge \text{LimAvg}(y) < 2$

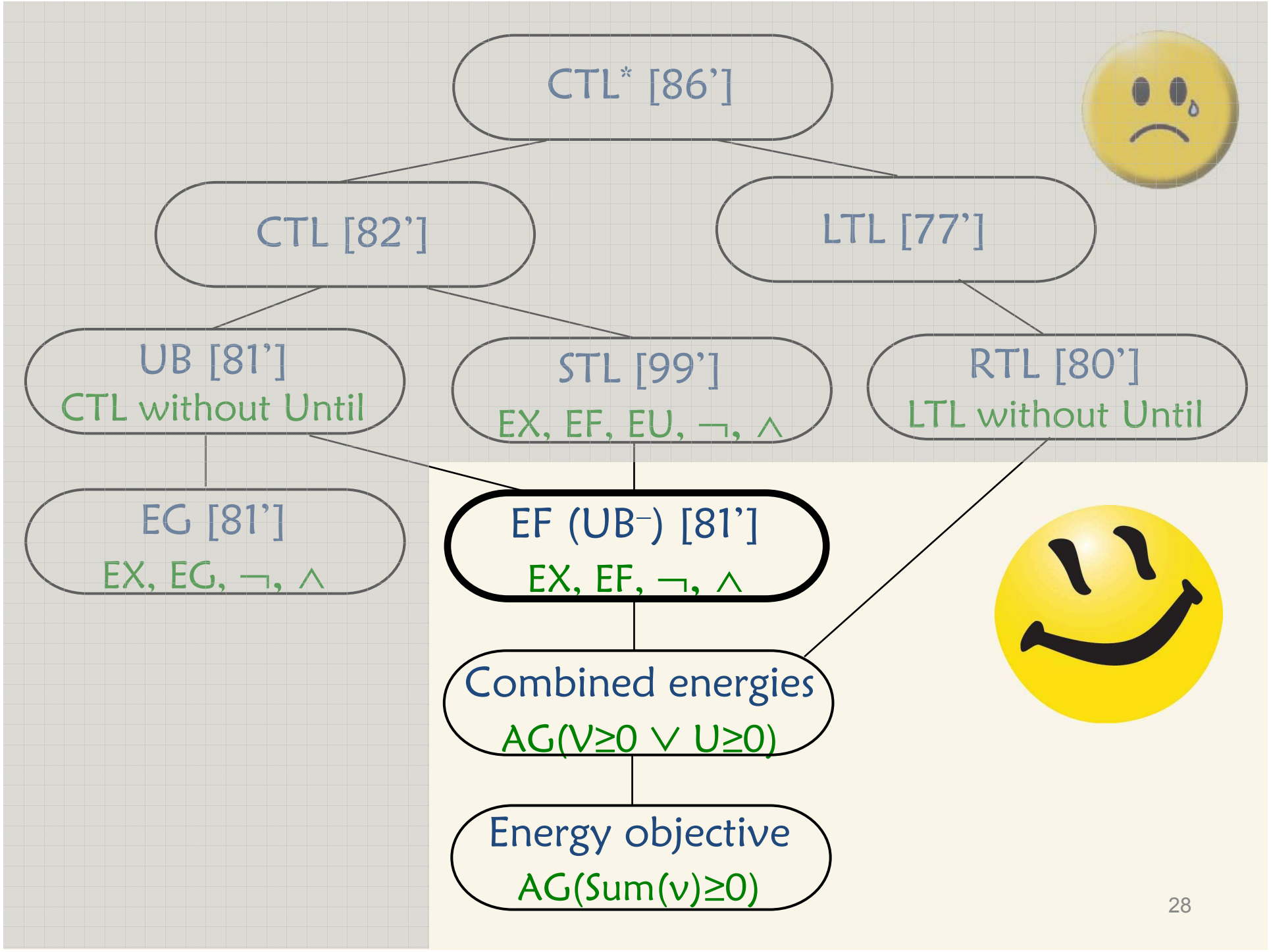
Is it decidable?

Yes!

Proofs

Technical details

Proofs of Part 1


CTL* [86']
CTL [82']
LTL [77']
UB [81']
CTL without Until
STL [99']
EX, EF, EU, ¬, ∧
RTL [80']
LTL without Until
EG [81']
EX, EG, ¬, ∧
EF (UB⁻) [81']
EX, EF, ¬, ∧
Combined energies
AG(V≥0 ∨ U≥0)
Energy objective
AG(Sum(v)≥0)

Part 1 - Proofs

- Udecidability proof
  - Reduction of the counter-machine halting problem.
  - Small adaptations to a Petri-net proof [Esparza 95].
  - Nice and simple.
- Decidability proof
  - Formulating the model checking problem in Presburger arithmetic.
  - Nice, … less simple.

Counter Machines

- A sequence of uniquely-labeled commands.
- Five command types, as in the following example.

L_1. if x=0 then goto l_5 else goto l_2

L_2. x := x-1

L_3. y := y+1

L_4. goto l_1

L_5. halt

  - The above counter machine uses two counters, x and y. It adds the value of x to y and nullifies x.

Counter Machines → Kripke Structure

- Straightforward, … except for the conditional jump.

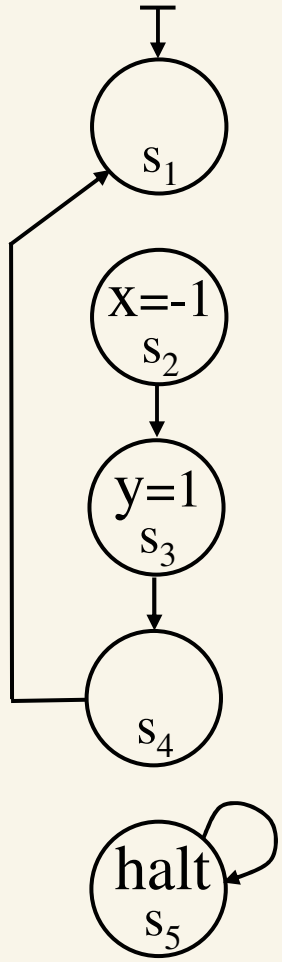
L_1. if x=0 then l_5 else l_2

L_2. x := x-1

L_3. y := y+1

L_4. goto l_1

L_5. halt



- A state for every line;
  a variable for each counter.

Conditional Jump ≠ Nondeterminism

- A naïve try would be to use nondeterminism.

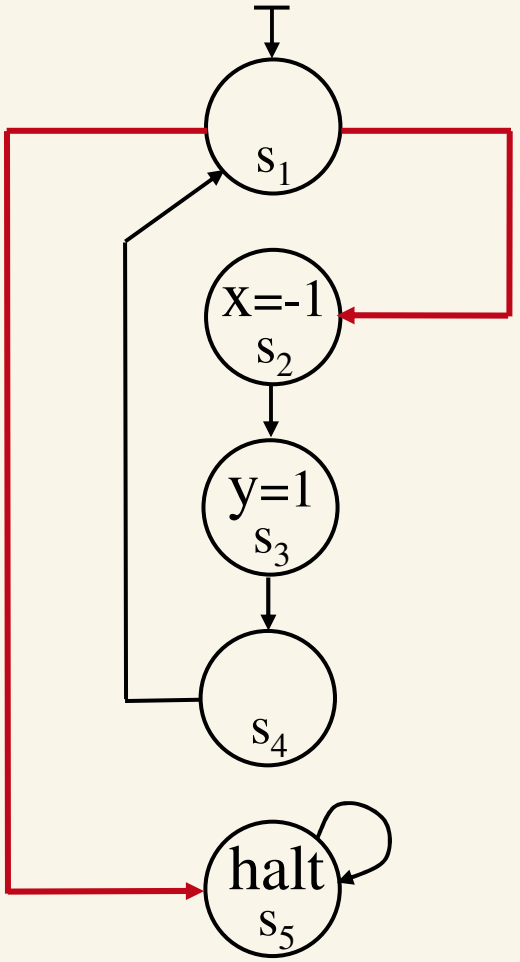
L_1. if x=0 then l_5 else l_2

L_2. x := x-1

L_3. y := y+1

L_4. goto l_1

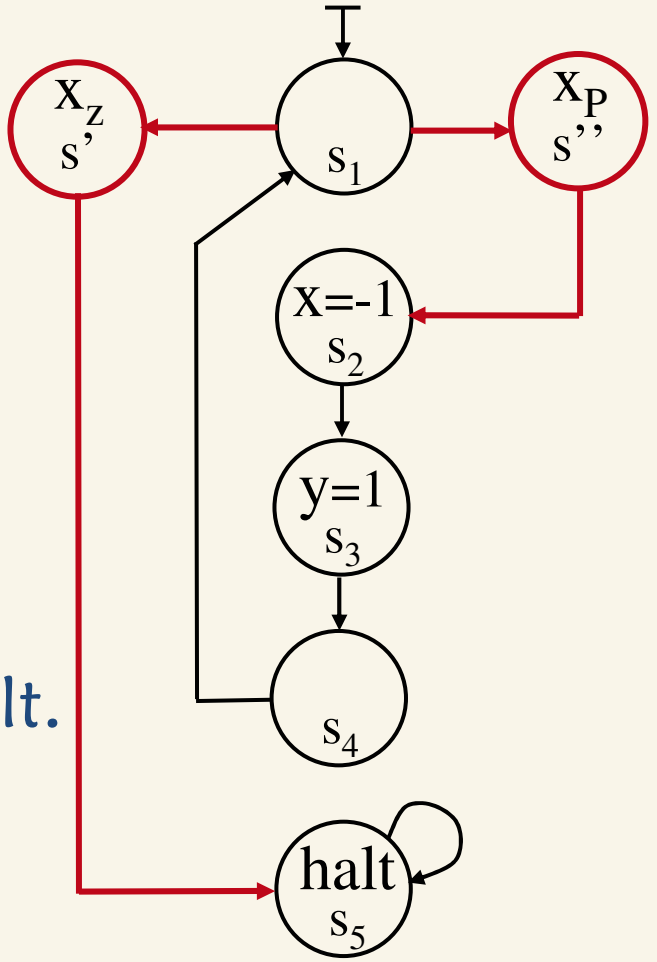
L_5. halt



- Which is, of course, wrong.

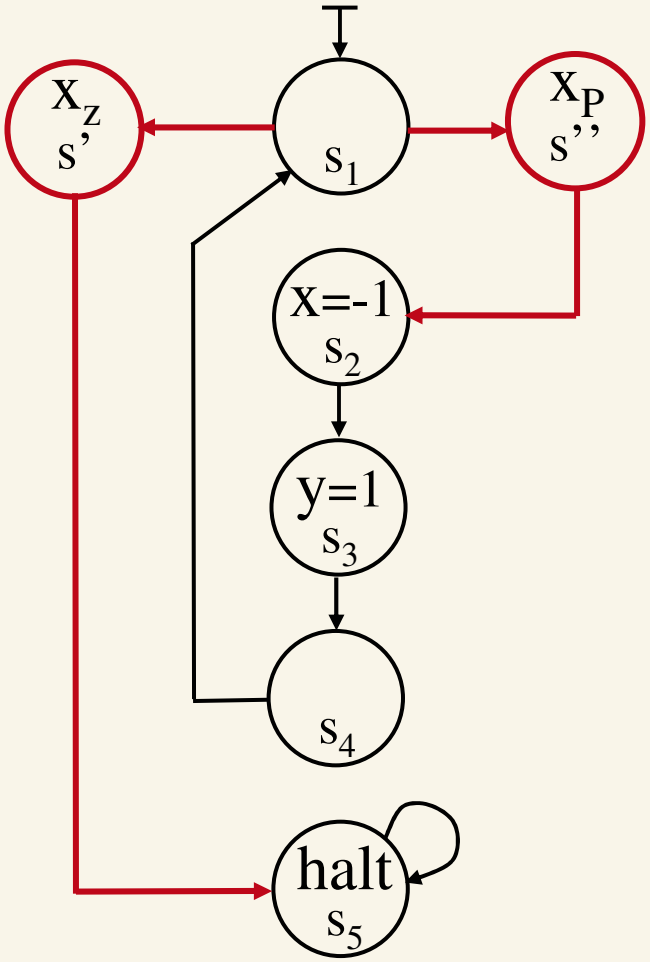
Conditional Jump = Nondet. + Check

- We only need a specification for a run that always makes the proper choices.
  - Proper: $\neg x_z \vee \text{Sum}(x)=0$ $\wedge \neg x_p \vee \text{Sum}(x)\neq 0$
  - Spec: EG(Proper $\wedge \neg$halt)
- The Kripke structure satisfies the specification iff the counter machine does not halt.

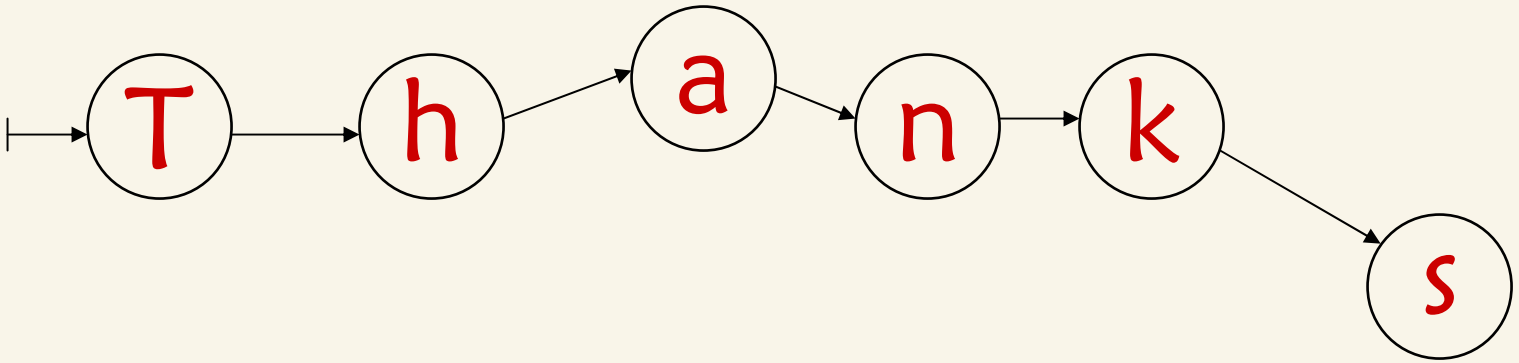


Other Temporal Operators

- [R]elease and [W]eak-until directly imply [G]lobally.
  - EG(p) = p EW False
    = False ER p
- EU does not imply EG.
  Yet, it is undecidable by the following specification.
  - proper: $\neg x_z \vee Sum(x)=0$
    $\wedge \neg x_p \vee Sum(x)\neq 0$
  - Spec: proper EU halt
- The Kripke structure satisfies the specification iff the counter machine halts.



GF(Talks end)



Decidability of the EF Logic (EX, EF, $\neg$, $\wedge$)

Presburger Arithmetic (PA)

- The first-order theory of natural numbers with addition. It has constants 0 and 1 and a binary function +.

  Axioms:

  - $\neg(0 = x + 1)$
  - $(x + 1 = y + 1) \rightarrow x = y$
  - $x + 0 = x$
  - $(x + y) + 1 = x + (y + 1)$

  Induction scheme:

  - For every formula $P(x)$: $(P(0) \wedge \forall x(P(x) \rightarrow P(x + 1))) \rightarrow P(y)$

- The theory is consistent, complete and decidable [Presburger 1929].

Model Checking by PA-Formulation

1. Move the Kripke-structure values to the edges.
2. Define a PA-variable x_i for every edge e_i.
   It will stand for the number of times that this edge is taken in a (finite) run that satisfies EF(p).
3. Naïvely formulate the specification in PA.
4. Formulate the "segments" of the Kripke structure.
5. Top to bottom, split the formula into a disjunction of sub-formulas, each w.r.t. a specific segment.

Proofs of Part II

Model-Checking LTL with LimAvg

- Negate the specification, and solve "exists a run s.t."
- A LimAvg proposition relates to the run's suffix. Hence, it has the same truth value in all positions.
- Therefore, we can split the specification into a disjunction of formulas, each having a specific truth value assignments to the LimAvg propositions.
- Solve the conjunction of LTL and LimAvg propositions, by solving the emptiness problem of a fair quan. Kripke structure with LimAvg objectives.
- For the latter, use convex-hull manipulations, along the lines of [ADMW09].

[ADMW09] "On Omega-Languages Defined by Mean-Payoff Conditions"
by Alur, Degorre, Maler and Weiss

GF(Talks end)