Bringing Asynchrony to Reaction Systems (work in pre-progress!)

Alex Yakovlev

Newcastle University

Newcastle upon Tyne, U.K.

Collaboration with Maciej Koutny

Workshop on Reaction Systems, Toruń, Poland, 7th June 2019

Abstract

Reaction systems have a number of underlining principles that govern them in their operation. They are: (i) maximum concurrency, (ii) complete renewal of state (no permanency), (iii) both promotion and inhibition, (iv) 0/1 (binary) resource availability, (v) no contention between resources. Most of these principles could be seen as constraints in a traditional asynchronous behaviour setting. However, under a certain viewpoint these principles do not contradict to principles underpinning asynchronous circuits if the latter were considered at an appropriate level of abstraction. Asynchrony typically allows enabled actions to execute in either order, retains the state of enabled actions while other actions are executed, involves fine grained causality between elementary events and permits arbitration for shared resources. This talk will discuss some of these potential controversies and attempt to show ways of resolving them and thereby bringing asynchrony into the realm of reaction systems. Besides that, we will also look at how the paradigm of reaction systems can be exploited in designing concurrent electronic systems.

Outline

- **Asynchronous Design Principles – quick review**
- **Reaction Systems: Basic Principles**
- **Asynchronous behaviour assumptions**
- **How do Reaction Systems meet them? Are Reaction Systems already asynchronous?**
- **What is Time in Reaction Systems? Model of delay?**
- **Possible ways of desynchronizing RS**
- **Examples of asynchronous behaviour of RS**

Key Principles of Asynchronous Design reviewed

- Asynchronous handshaking – *control is distributed*
- Delay-insensitive encoding – *data carries validity tags*
- Completion detection – *all operations must produce done signals*
- Causal acknowledgment (aka indication or indicatability) – *precedence is governed by explicit cause-effect relations*
- Strong and weak causality (full indication and early evaluation) – *precedence can have different modality*
- “Time comparison” (synchronisation, arbitration) – *conflicts and non-determinism requires explicit decision elements*

Reaction systems: basic principles

- (RS1) maximum concurrency,
- (RS2) complete renewal of state (no permanency),
- (RS3) both promotion and inhibition,
- (RS4) 0/1 (binary) resource availability (no counting),
- (RS5) no contention between resources.

From GR lectures:

NO PERMANENCY:
*An element of **T** vanishes unless it is sustained by a reaction!*

This reflects the basic bioenergetics of the living cell: without the flow/supply of energy the living cell disintegrates.

But energy use/absorption is a chemical process achieved through biochemical reactions.

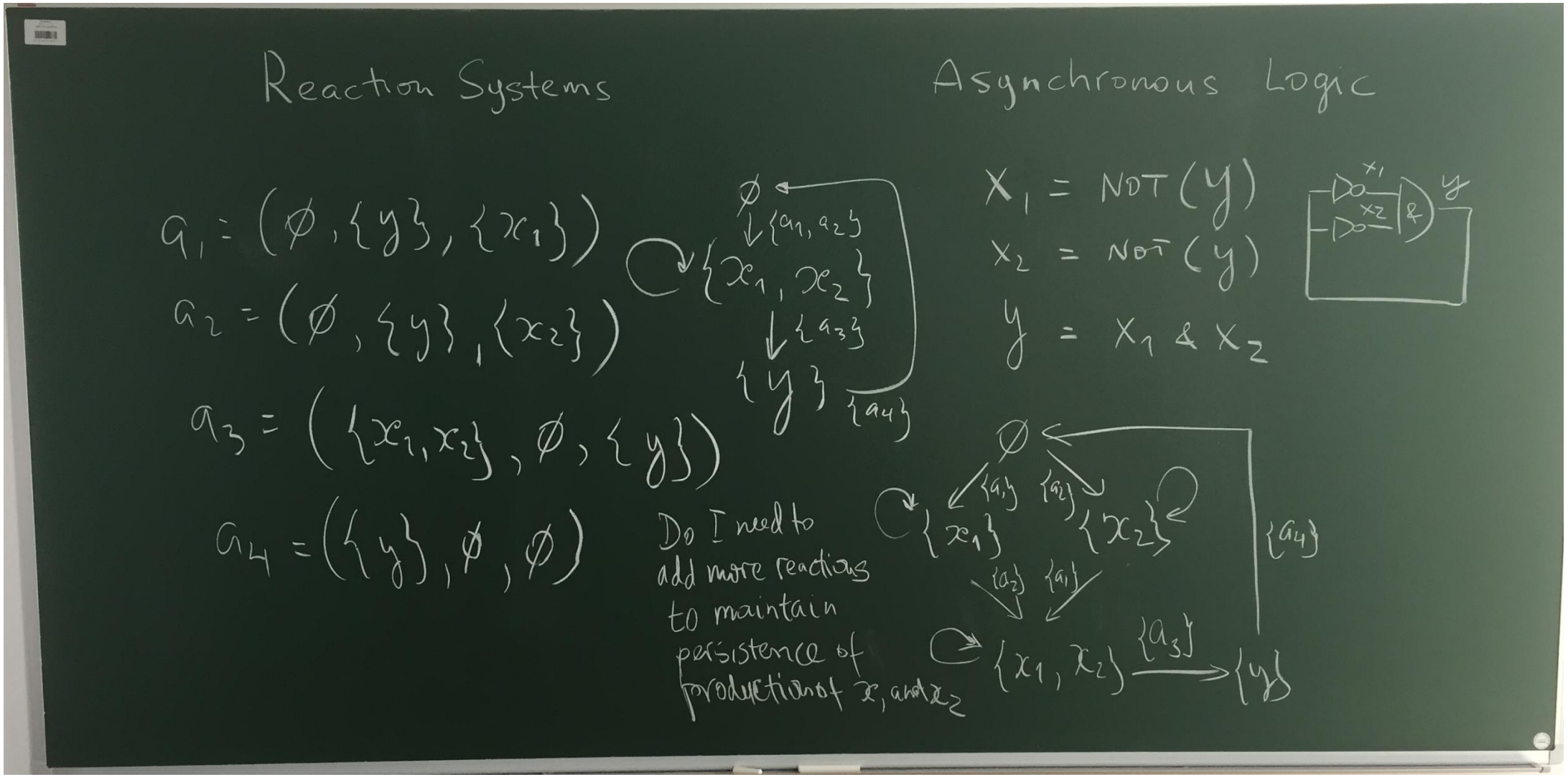
From GR lectures:

THRESHOLD NATURE OF RESOURCES
either something is available and there is enough of it
or it is not available
Hence: NO COUNTING !!!

The basic model is qualitative.

This reflects the level of abstraction we adopted for the formulation of our basic model.

Reaction system dynamics

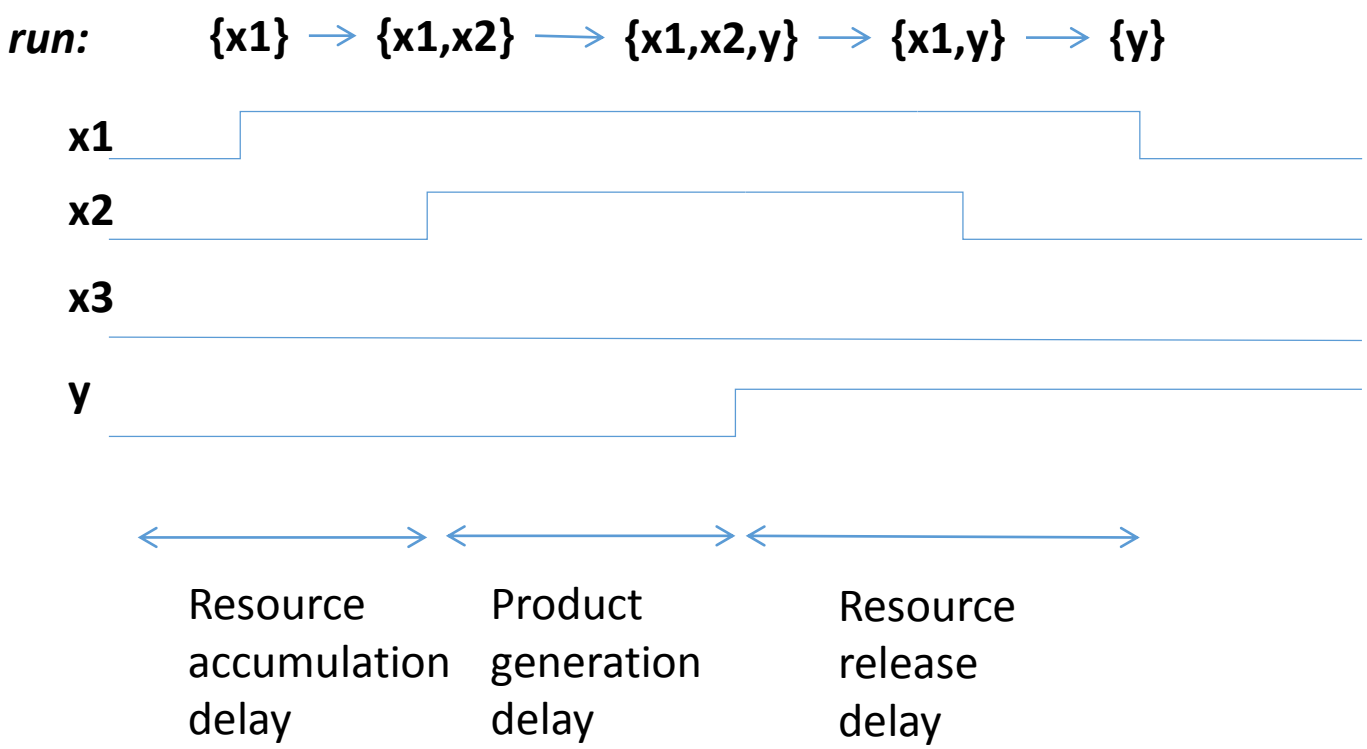


Important attributes of the Model

- What is the model of delay of a reaction?
  - Example in Muller's circuits: Each logic element consist of: Functional Evaluator, which is instantaneous, plus a Finite (arbirary) inertial Delay
- What is the notion of correct behaviour in the model, so we can capture "bad behaviour" – e.g. properties of the model may correspond to undesirable behaviour in the physical system?
  - Example in Muller's circuits: An enabled logic element cannot be disabled except by firing (in Petri net models – we use either 1-safety or persistence conditions)

Model of delay

a: ({x1,x2}{x3}{y})



Other delay models are possible

Some good news in Reaction Systems for modelling asynchronous logic circuits

- No conflicts on reading resources: signals are easy to branch to as many logic elements as needed; cf. read-arcs for free!
- No counting on writing to resources: signals are naturally produced in an OR-causal way; no need to get rid of extract tokens

Some bad news for modelling asynchronous logic

- Resources that are read are consumed by reactions and require to be explicitly re-introduced which may cause problems with maintaining enabled reactions
- No simple way to resolve timing conflicts (for asynchronous time comparison); but this might still be possible with additional resources and we may allow some quantitative comparisons (on concentrations – similar to analog amplifiers) of resources/products between reactions.

Challenge: finding an (elegant) way to express synchrony and asynchrony in RS

- Are Reactions Systems (fundamentally) (a)synchronous, i.e. self-timed?
- Are their existing interpretation (semantics) synchronous?
  - It seems that synchronous operation is easy to capture in the RS – e.g. processes
- So, is it all about finding the right modelling level to express asynchrony?

Asynchronous systems: basic assumptions/requirements

Asynchrony typically allows:

- (AS1) enabled actions may execute in either order (actions have their local timing mechanisms – e.g. delays),
- (AS2) the state of enabled actions is retained while other actions are executed,
- (AS3) fine-grained causality is allowed between elementary events (e.g. based on precedence),
- (AS4) arbitration for shared resources is permitted (e.g., explicit decision elements, arbiters)

Note: “dynamic” inhibition (disabling an enabled action) can only be performed via explicit arbitration – part of AS4

Research Questions

- (Q1) At what level of reaction systems can we apply asynchronous interaction? Can reaction systems be distributed in space and (hence) in time?
- Can we build reactions systems with some level of control/policy – cf. GALS systems in electronics
- (Q2) What determines causality between actions in electronic circuits?
  - (P1) Passage of control information (cause-effect paradigm)
  - (P2) Passage of material resources (produce-consume paradigm)
- Reaction systems appear to be fundamentally based on P2 paradigm
- Can we adopt this approach in computational processes, and build electronic systems with inherent resource-based causality? – cf. combining energy-information flow

Modelling possibilities

- Possibility 1: Introduce explicit delay model for a (generic) reaction; then properties of RS behaviours will form some classes: we will be able to distinguish good behaviours from bad behaviours (e.g. semi-modular or hazard-free RSs)
  - Plus: maintain the original set of resources and reactions
  - Minus: extra overhead – delay model
  - Examples at the end of the talk!
- Possibility 2: Model local interactions between reactions by explicit resources responsible for handshakes; all reactions interact with their neighbours by “handshake resources”
  - Plus: simplicity of structuring; correctness can be established by construction
  - Minus: extra overhead – we need many auxiliary resources

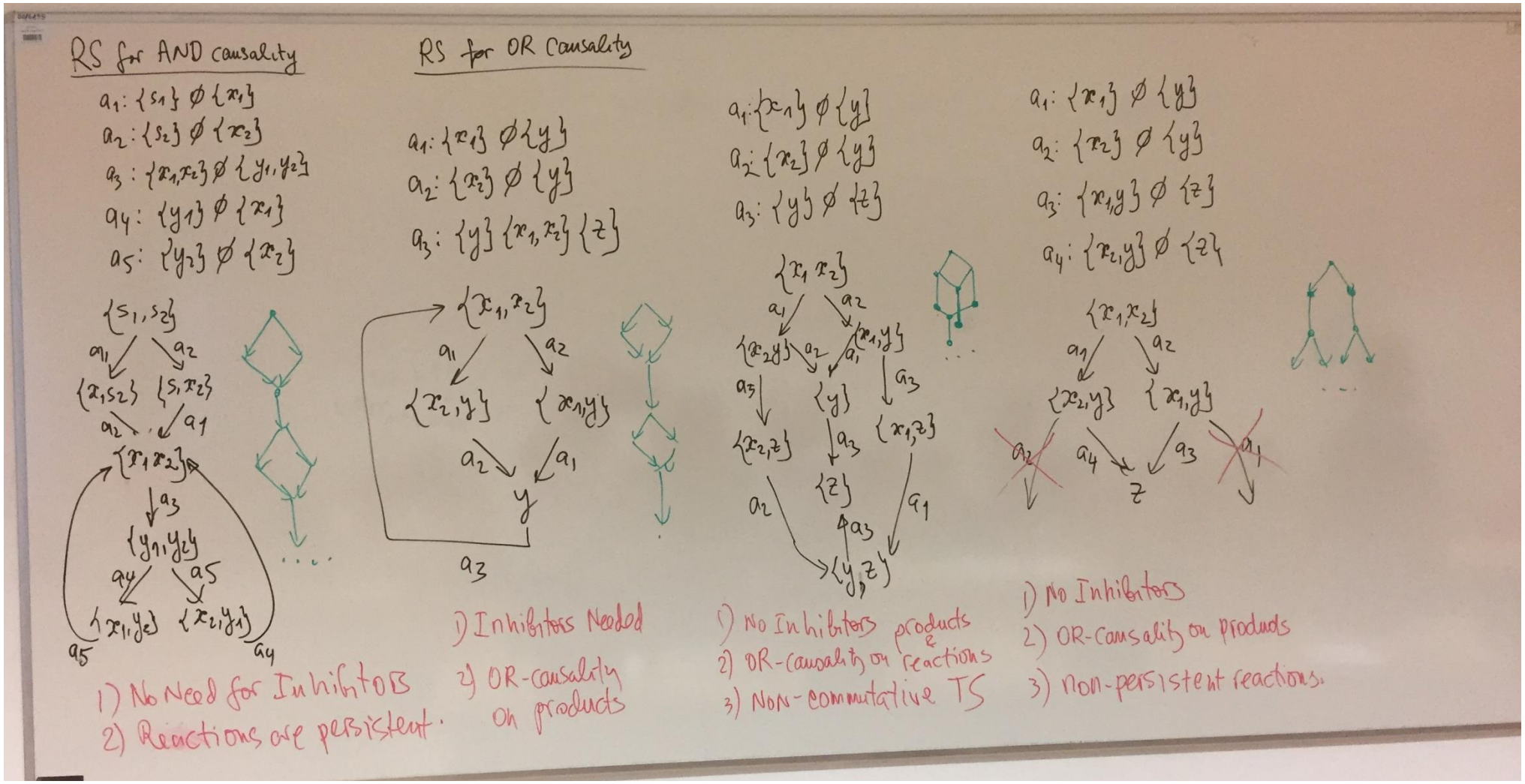
Modelling possibilities

- Possibility 3: Model some reactions as local clock generators; some reactants will be produced which will remain valid for the duration of all enabled reactions in the give locality (e.g. similar to GALS systems with start/stoppable clocks)
  - Plus: simplicity of structuring; correctness can be established by construction
  - Minus: extra overhead on clocking reactions, mechanisms for starting and stopping to be defined – possibly some handshake resources; still synchrony within the localities.
- Possibility 4: Latency insensitive (Elastic) reaction systems (still globally clocked); resources will be maintained, and produced by reactions for certain undefined (but finite) time intervals; separate clocking will be needed
  - Plus: simplicity – this method is probably already in Reaction Systems
  - Minus: global clocking (external to the Reaction System)

Examples of asynchronous behaviour in RS (white board) – for possibility 1

- Modelling AND (strong) and OR (weak) causality
- Behavioural classes of RS: with or without Inhibitors
- Properties:
  - Causality on reactions or on products (reactants) or both
  - Persistence and Commutativity of transition systems
  - Lattice properties on cumulative states (Parikh vectors on resources and reaction occurrences) – e.g. distributive, semi-modular
  - Delay-independence

Examples of asynchronous behaviour in RS



Conclusions

- Reaction systems can be seen in asynchronous way (self-timed)
- Delay models are essential
- Various ways of modelling asynchronous behaviour are possible
- Interesting behavioral classes can be observed in self-timed RS
- More work needed …
  - E.g. what to do with non-persistent reactions – conflicts on reactions but not resources are possible -> we need some “units” that compare reactant concentrations (quantitative?!) and produce the result at the RS level

- THANK YOU!