Coordinating Data-Parallel **SAC** Programs with **S-Net**

Sven-Bodo Scholz
Clemens Grelck and Alex Shafarenko

Compiler Technology and Computer Architecture Group
University of Hertfordshire, UK

HIPS-ToPMoDRS,
26.March 2007, Long Beach

www.aether-ist.org

SAC in a nutshell

- generic array programming
- purely functional
- no-frills
- aggressive program optimisation
- sequential performance competitive with FORTRAN77
- generation of multi-threaded code without annotations

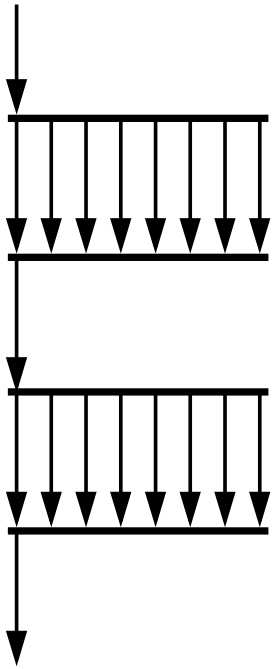
For details see www.sac-home.org

Exploitation of Data-Parallelism in **SAC**

- hinges on our data-parallel construct, the With-Loop
- all our array operations are defined by these
- granularity control by
  - array size inference
  - several fusion techniques
- architecture specific code generation
- POSIX-threads for SMPs
- taylor-made memory management
- almost linear speedups for examples such as NAS-benchmarks

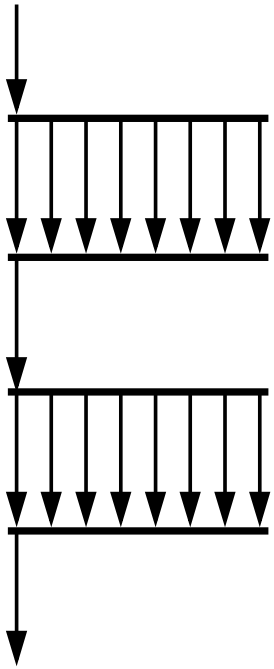
So what is the catch??

Our compiler maps all concurrency into one Concurrency pattern:



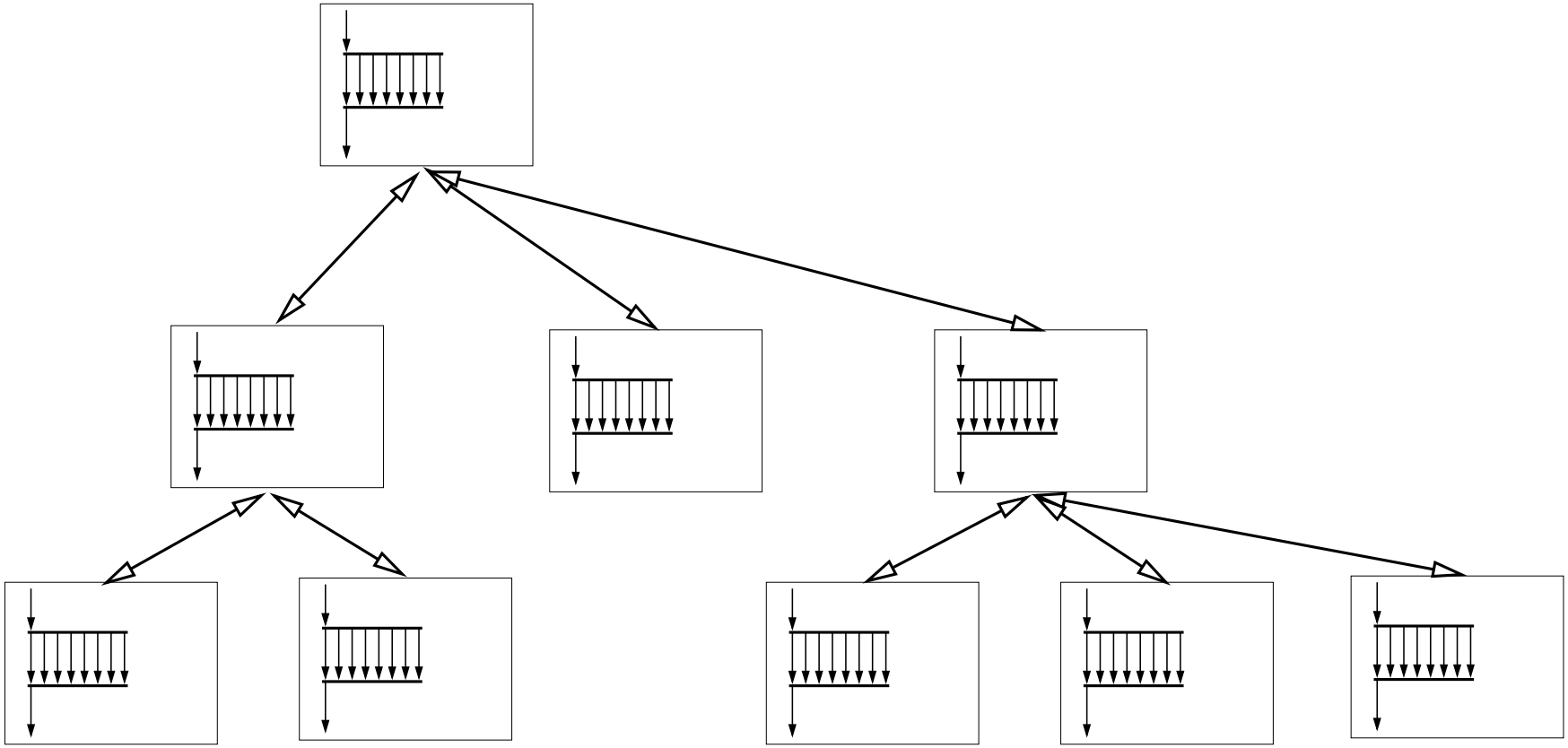
So what is the catch??

Our compiler maps all concurrency into one Concurrency pattern:



if our application cannot be mapped on this pattern, we cannot exploit its concurrency!

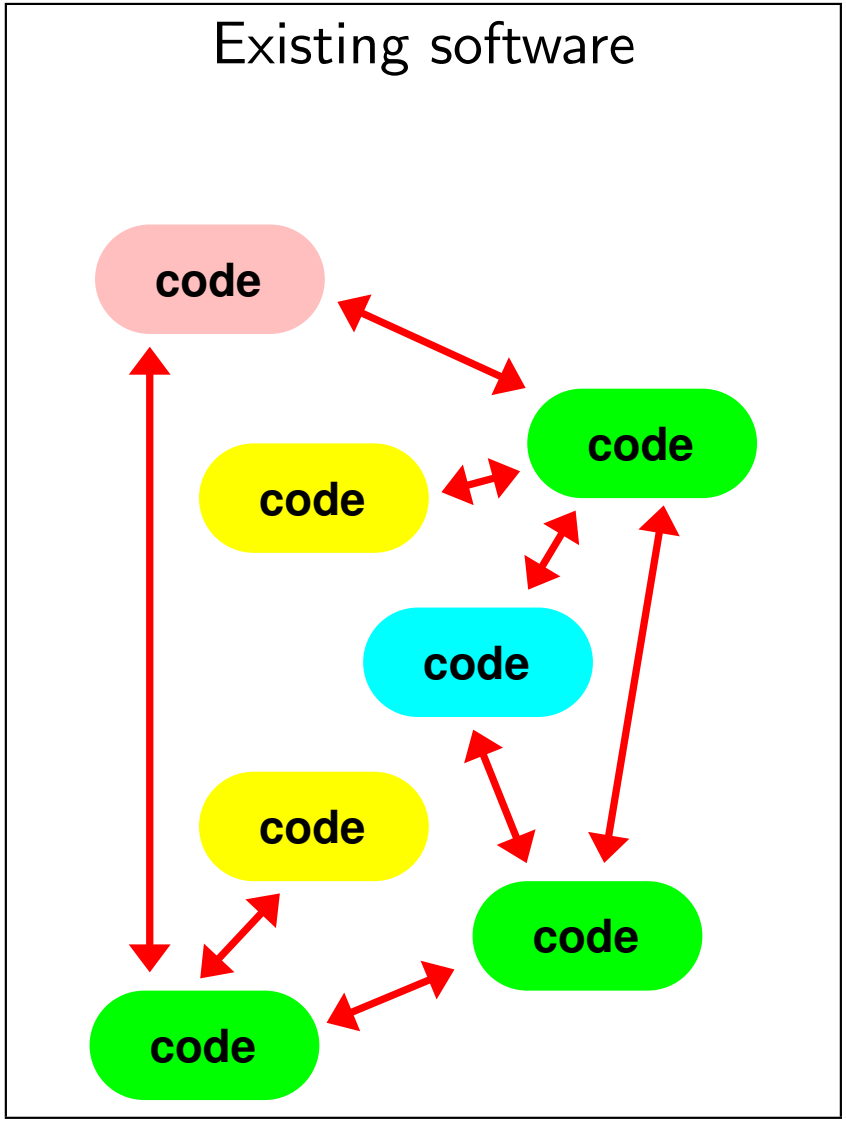
An example: Search Algorithms



Our Approach: **S-Net**

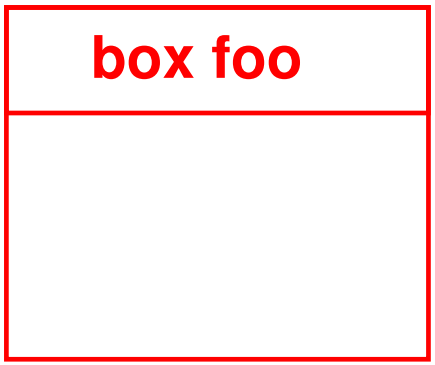
S-Net:

- asynchronous component framework
- functional stream processing
- pure coordination language
- dynamic network creation
- orthogonal to the box language



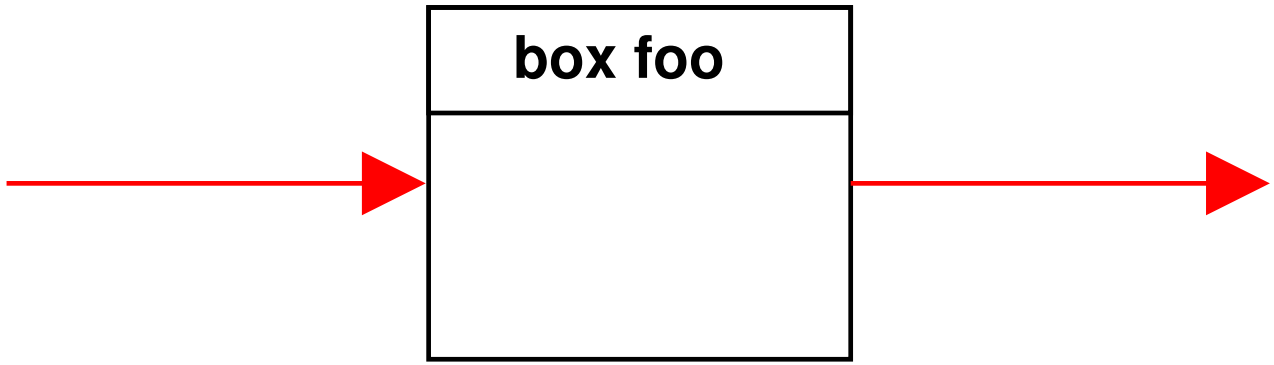
S-Net at a Glance

- central construct: box
  wraps existing code



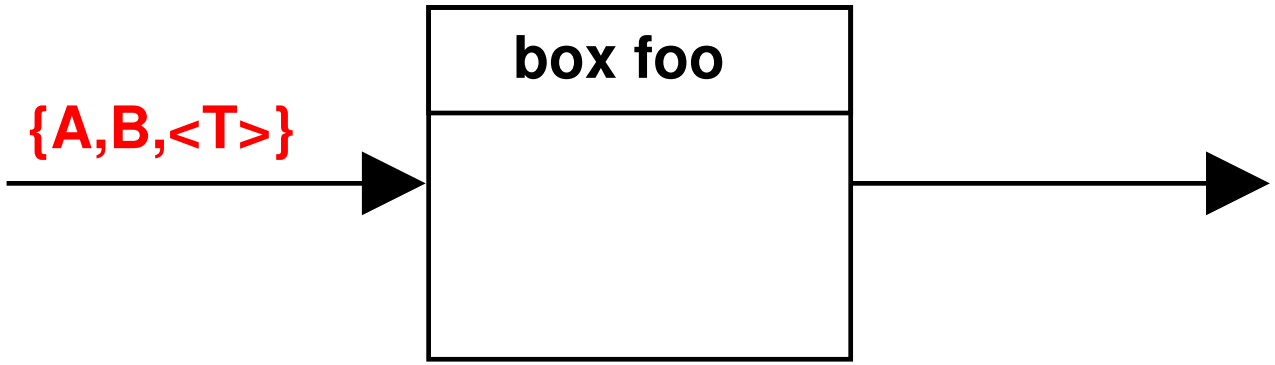
S-Net at a Glance

- central construct: box
  wraps existing code
- connected by
  - single input stream
  - single output stream



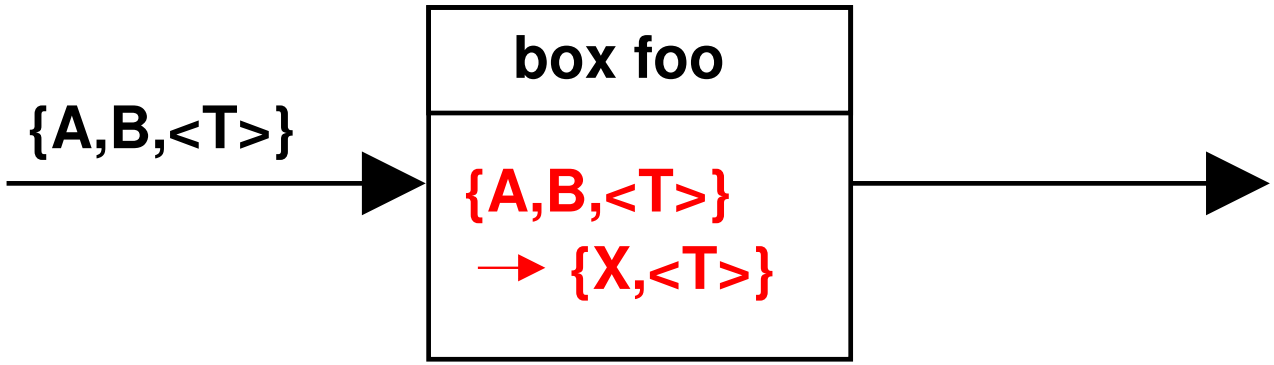
S-Net at a Glance

- central construct: box
  wraps existing code
- connected by
  - single input stream
  - single output stream
- streams transport records:
  sets of named fields
  - opaque value fields
  - integer-valued tag fields



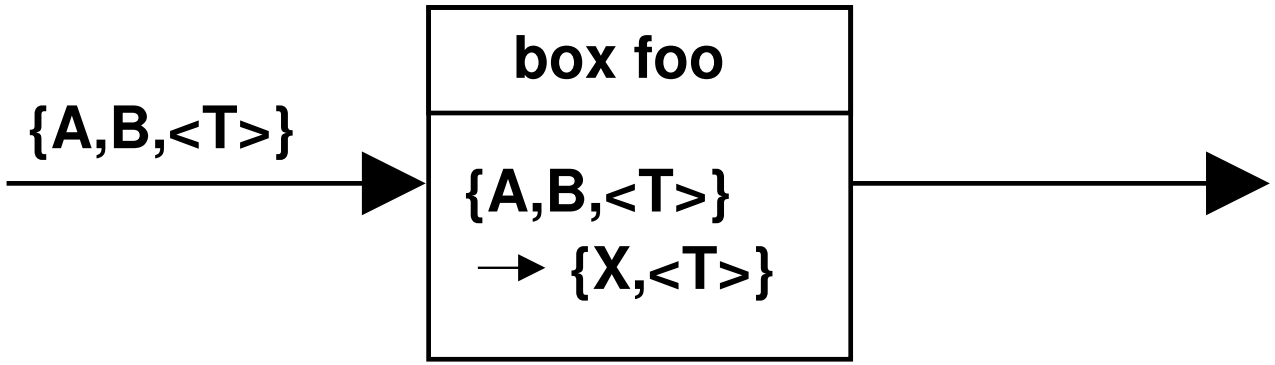
S-Net at a Glance

- central construct: box wraps existing code
- connected by
  - single input stream
  - single output stream
- streams transport records: sets of named fields
  - opaque value fields
  - integer-valued tag fields
- box behaviour declared by type signature



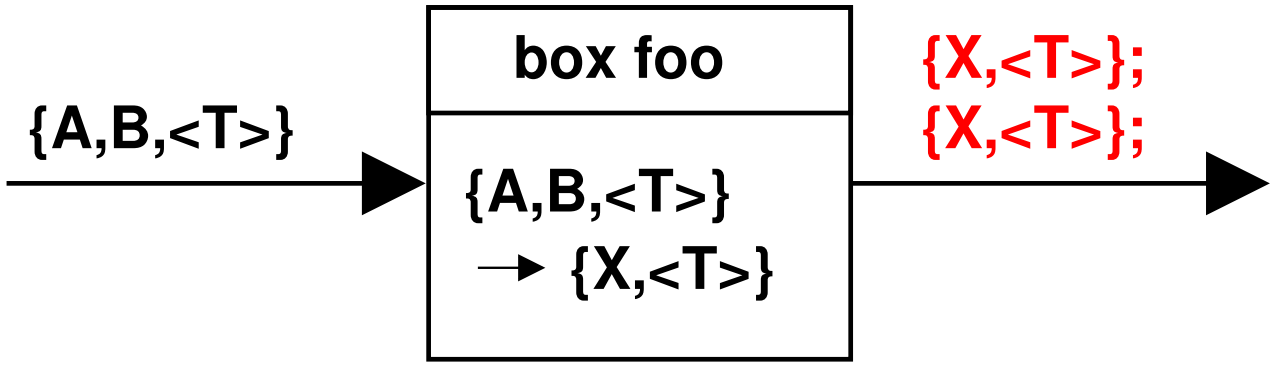
S-Net at a Glance

- central construct: box wraps existing code
- connected by
  - single input stream
  - single output stream
- streams transport records: sets of named fields
  - opaque value fields
  - integer-valued tag fields
- box behaviour declared by type signature
- behaviour defined in box language, not **S-Net**

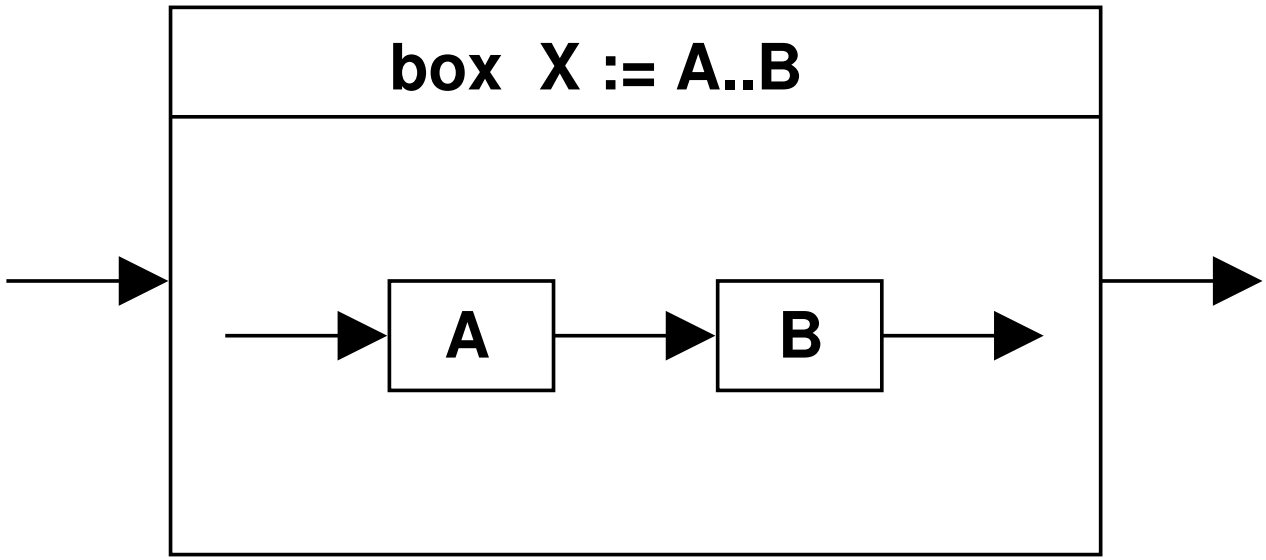


S-Net at a Glance

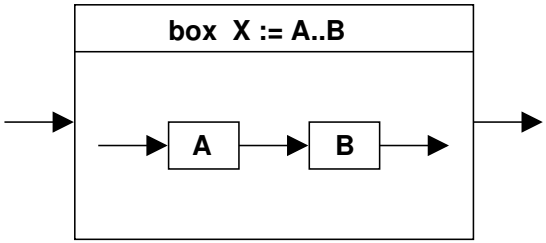
- central construct: box wraps existing code
- connected by
  - single input stream
  - single output stream
- streams transport records: sets of named fields
  - opaque value fields
  - integer-valued tag fields
- box behaviour declared by type signature
- behaviour defined in box language, not **S-Net**
- box maps single input record to stream of output records

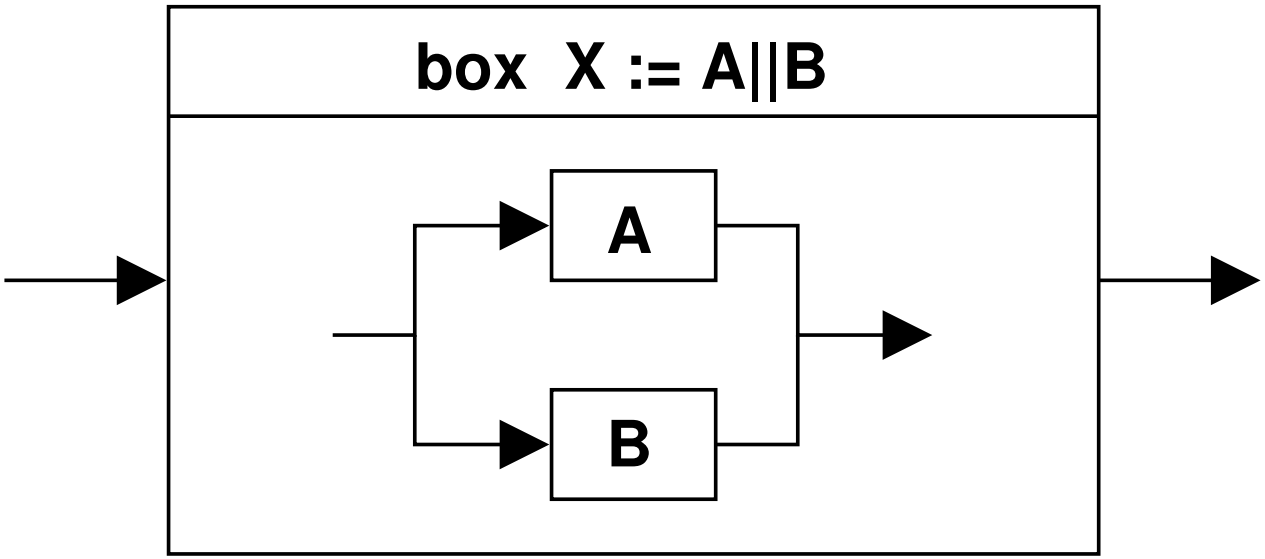


Network Combinators: Serial

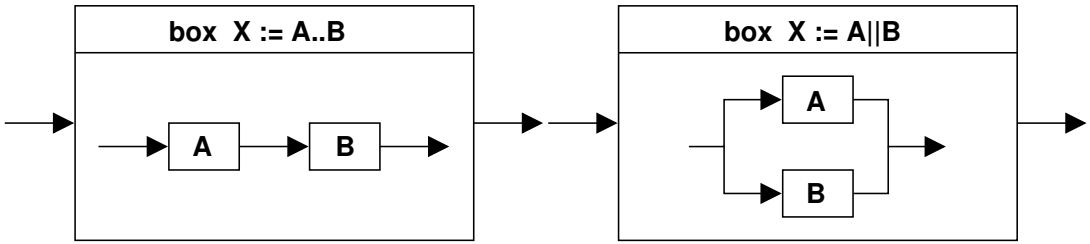


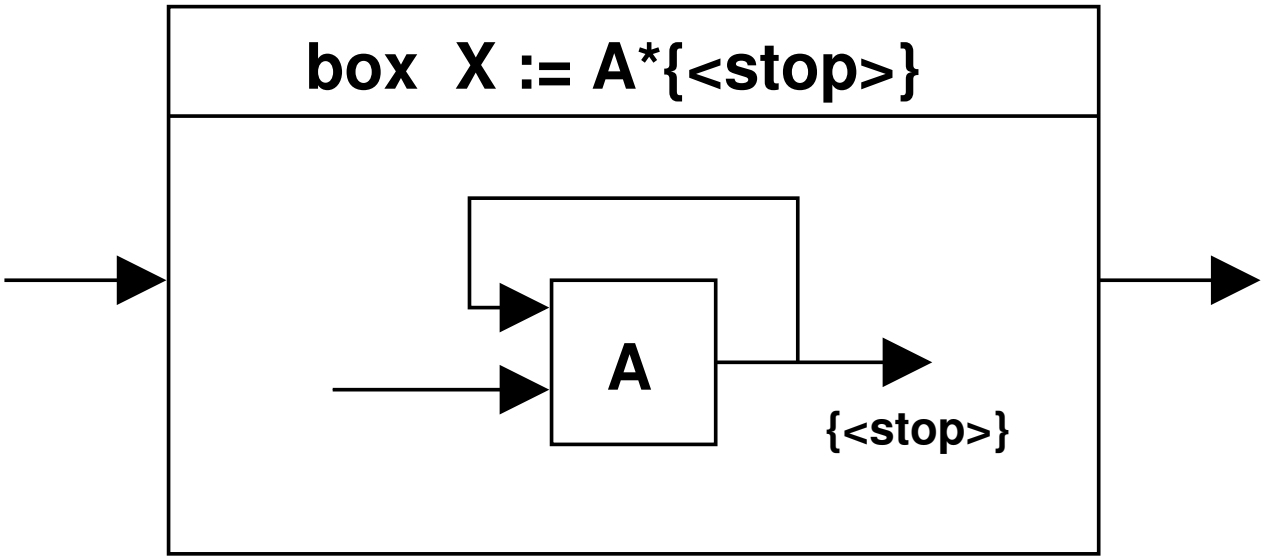
Network Combinators: Choice



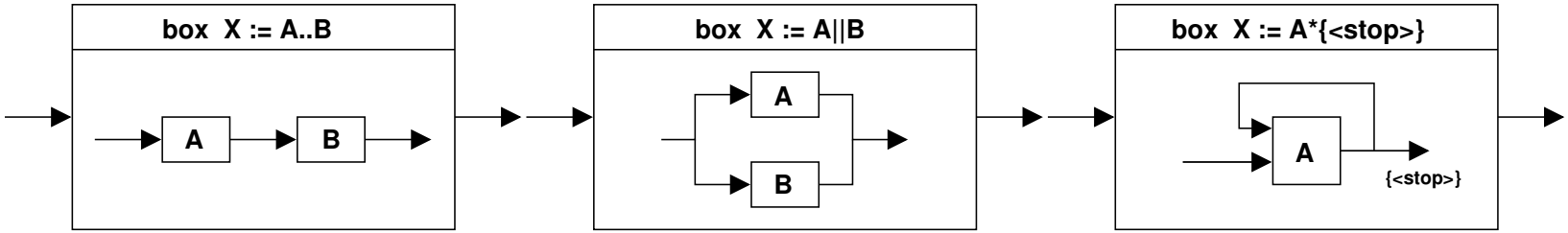


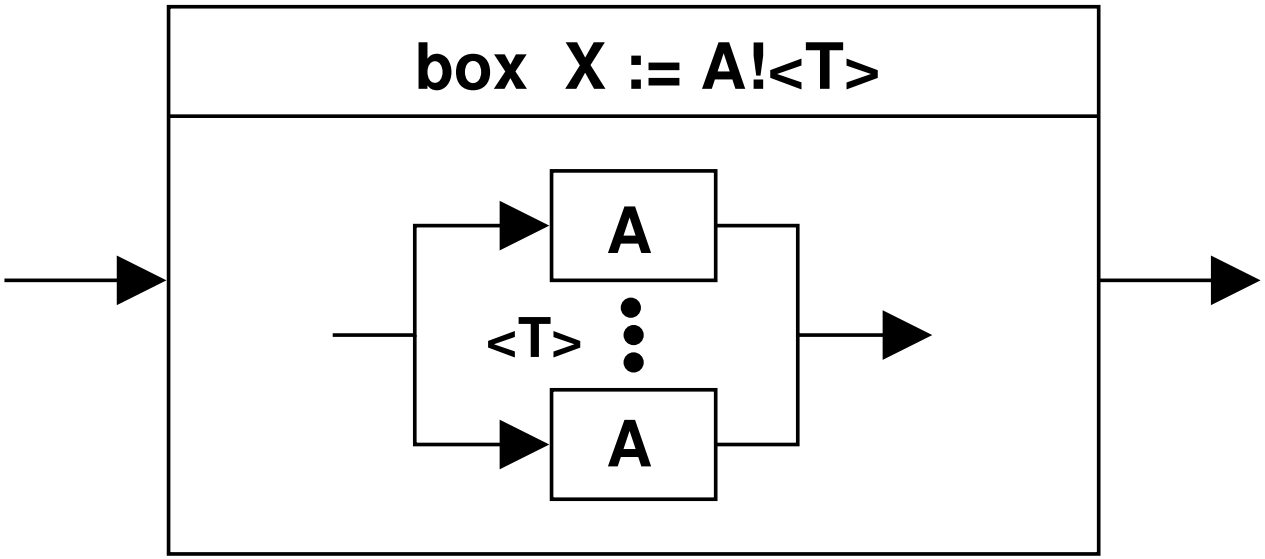
Network Combinators: Serial Replication



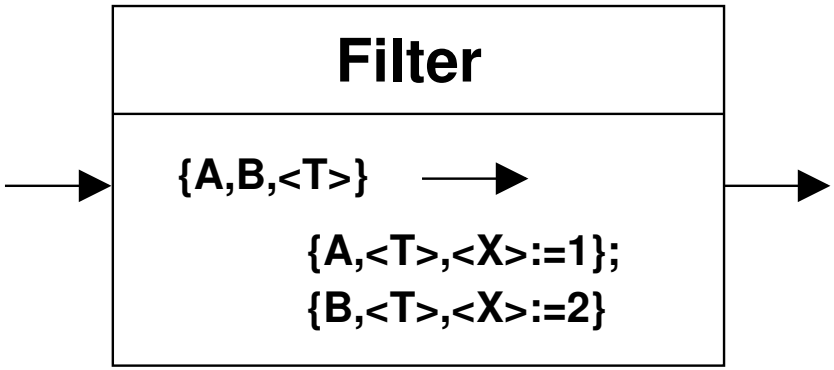


Network Combinators: Index Split





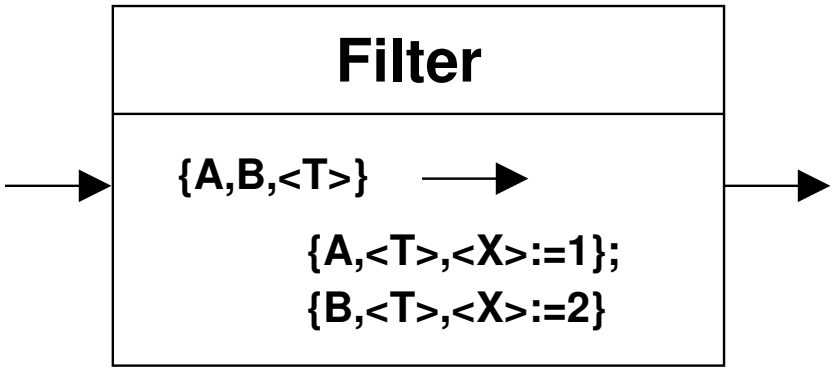
Primitive Boxes: Filter and Sync

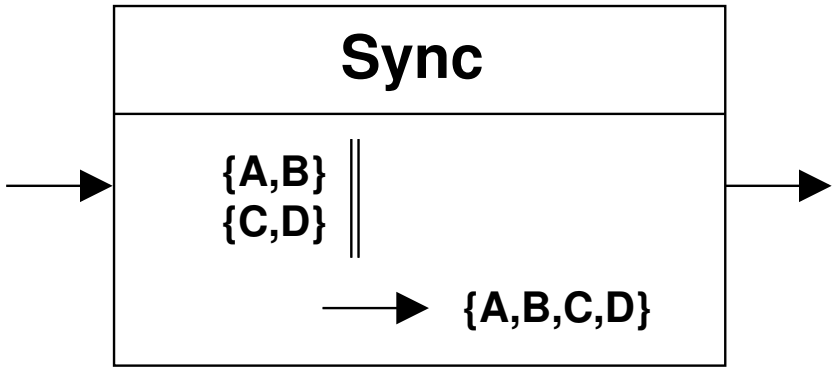


housekeeping:

- eliminate record fields
- duplicate record fields
- add tags
- manipulate tag values
- ...

Primitive Boxes: Filter and Sync





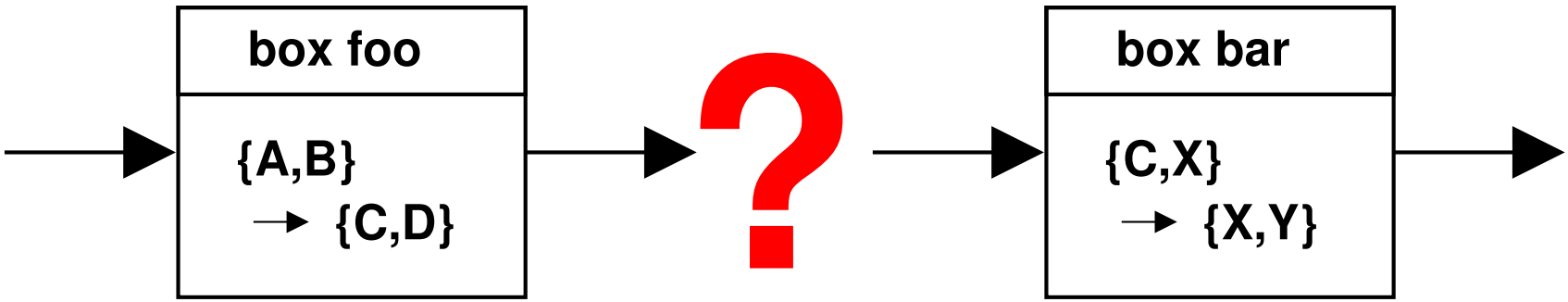
housekeeping:

- eliminate record fields
- duplicate record fields
- add tags
- manipulate tag values
- ...

synchronisation:

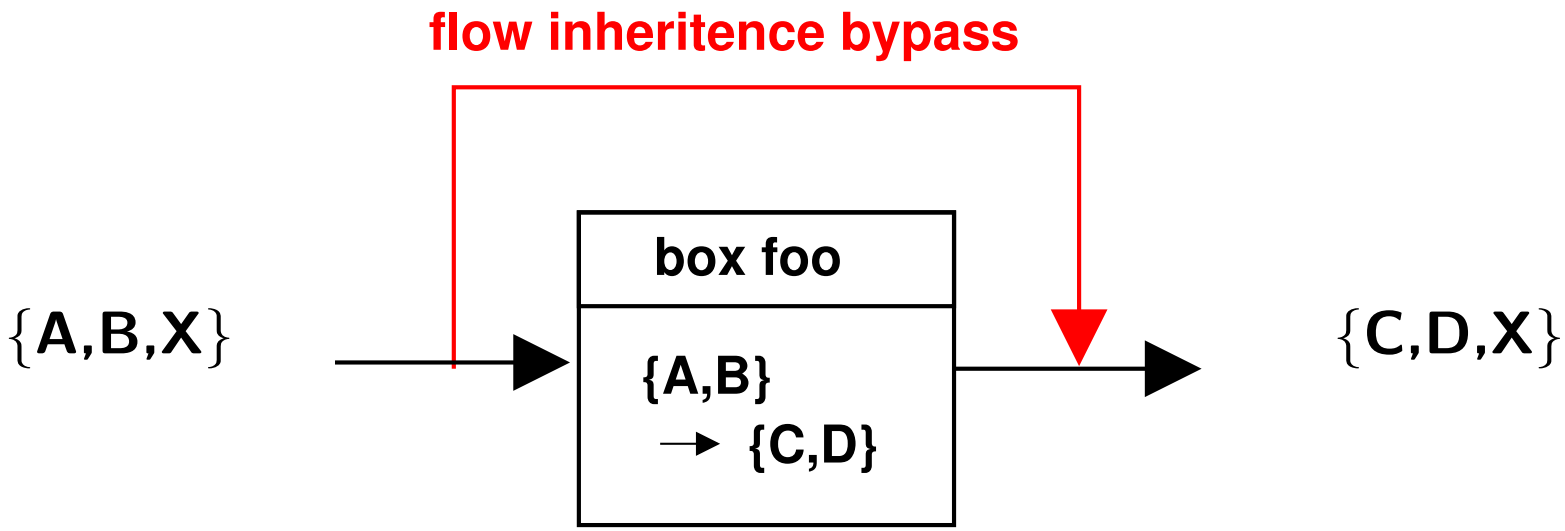
- keep record that matches pattern
- all patterns matched: release merged records
- pattern already matched: pass through

Problem: Incompatible Boxes



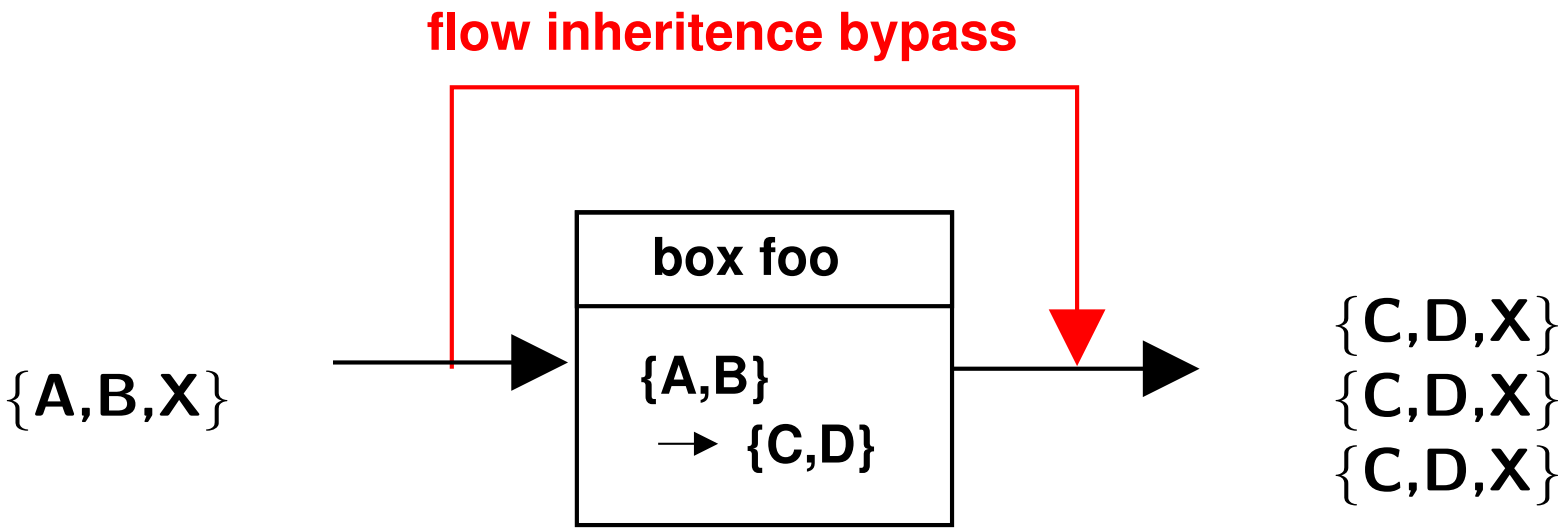
- Box code typically designed in isolation.
- Interfaces only partially overlap.
- Network composition unfeasible in practice?

Solution: Flow Inheritence



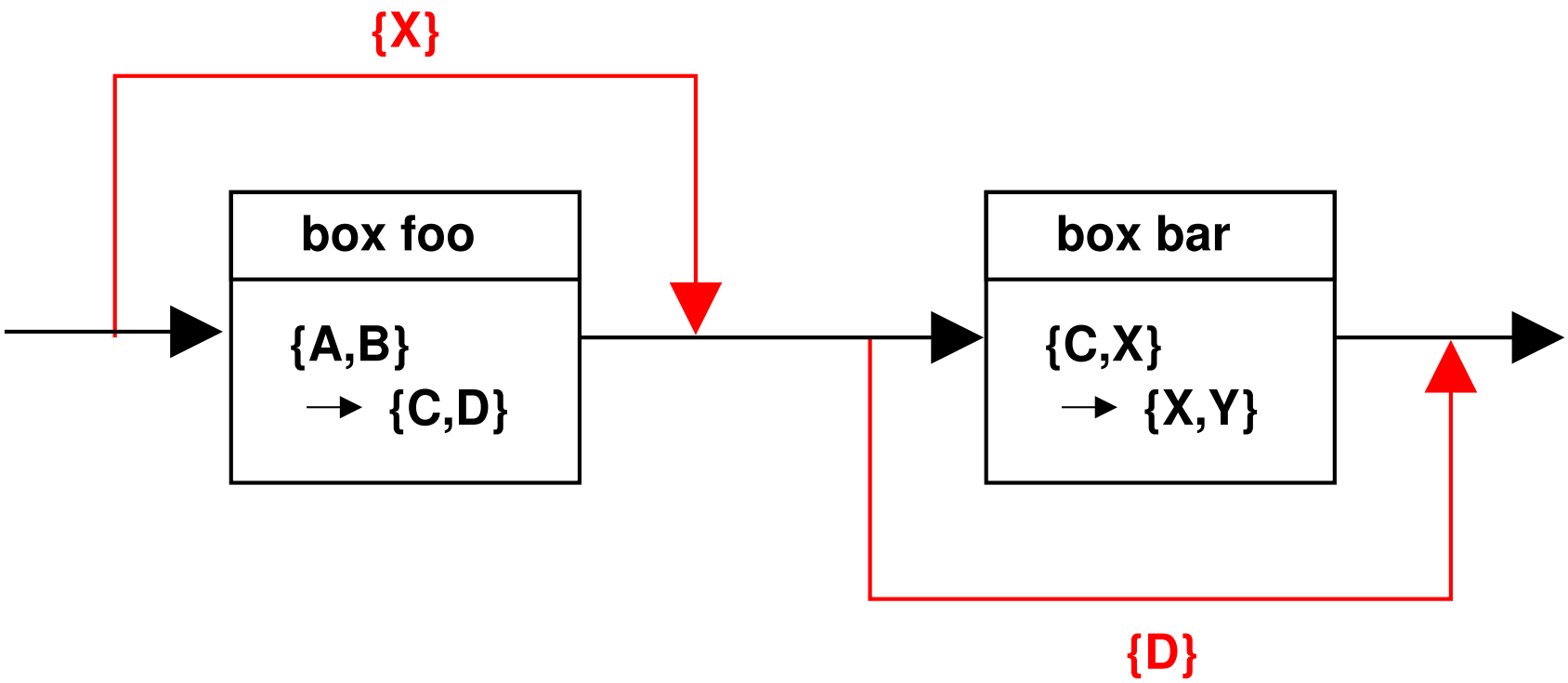
- Any "unwanted" field is bypassed to output channel
- and is attached to any record produced in response.

Solution: Flow Inheritence



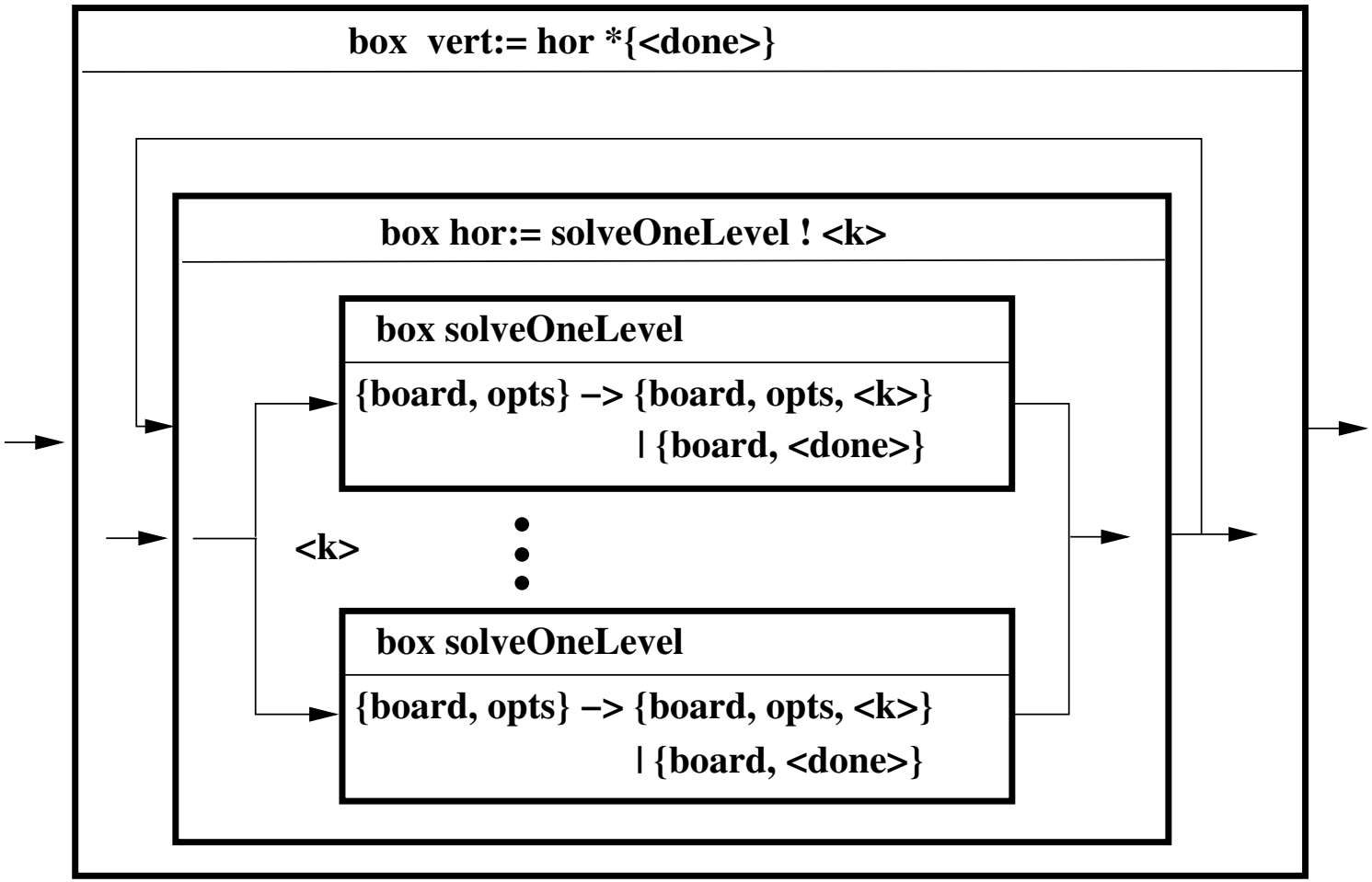
- Any "unwanted" field is bypassed to output channel
- and is attached to any record produced in response.

Flow Inheritence: Enabling Network Combination

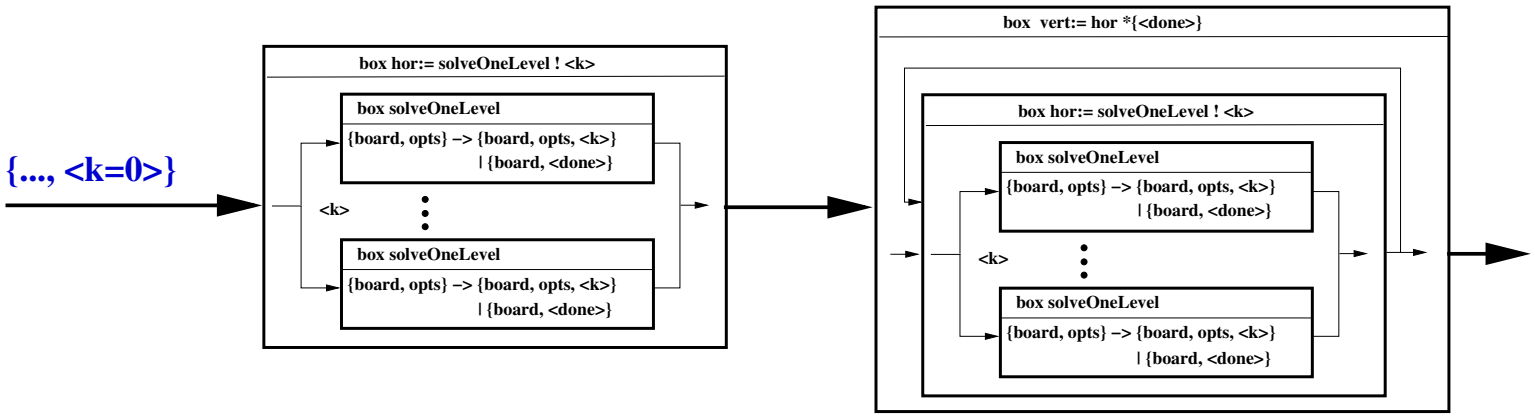


- Flow inheritence is the key to network glue.

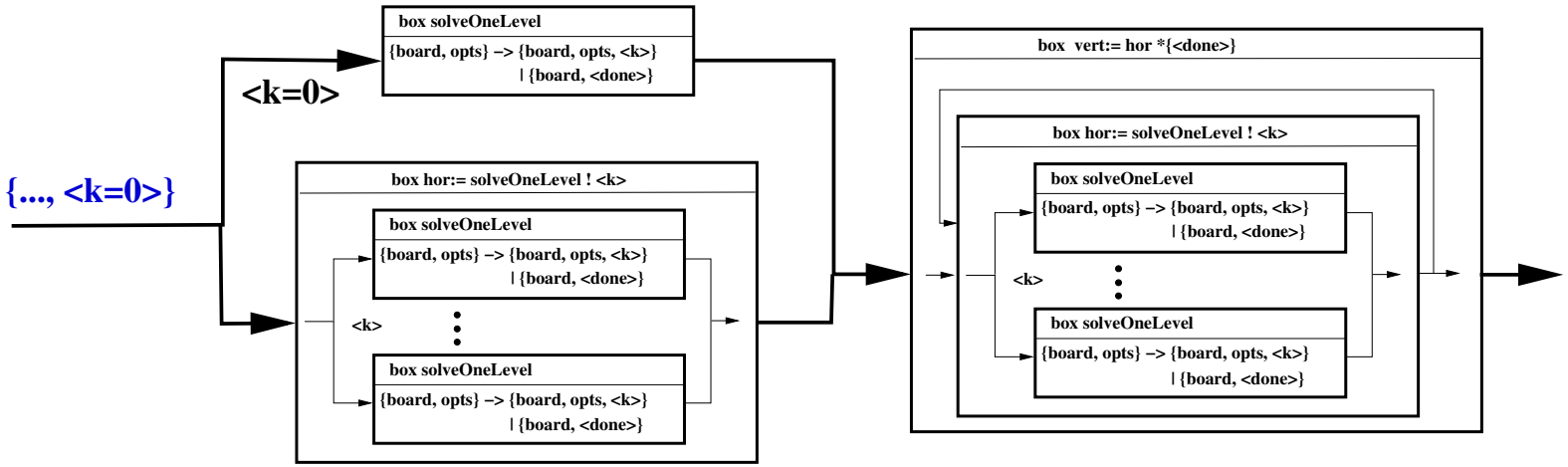
How to apply this to our problem?



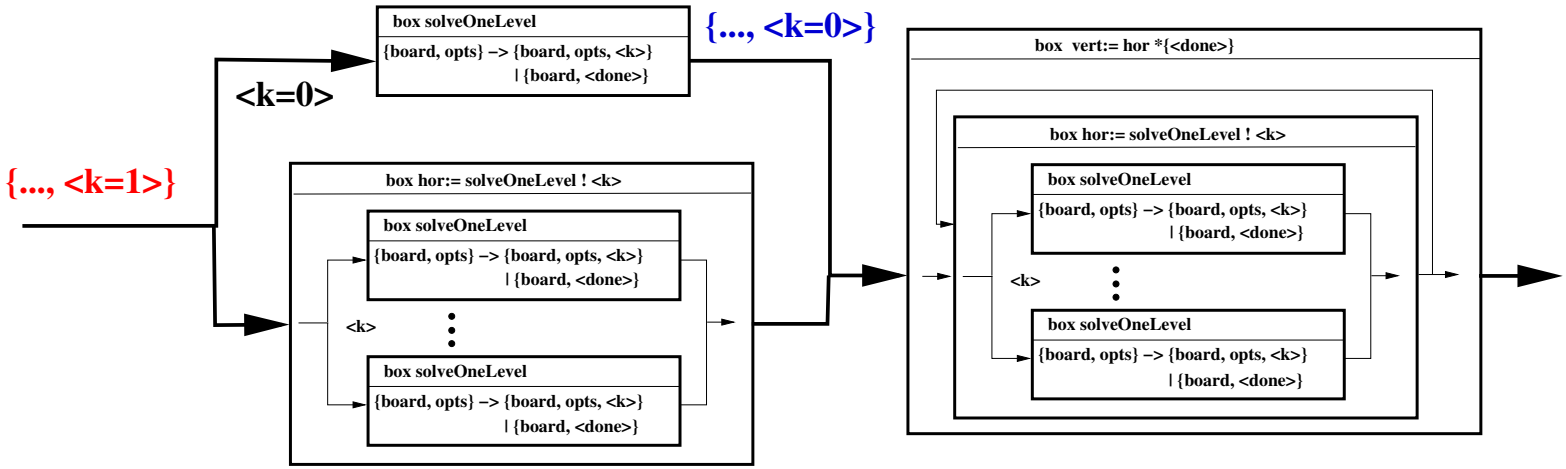
How to apply this to our problem?



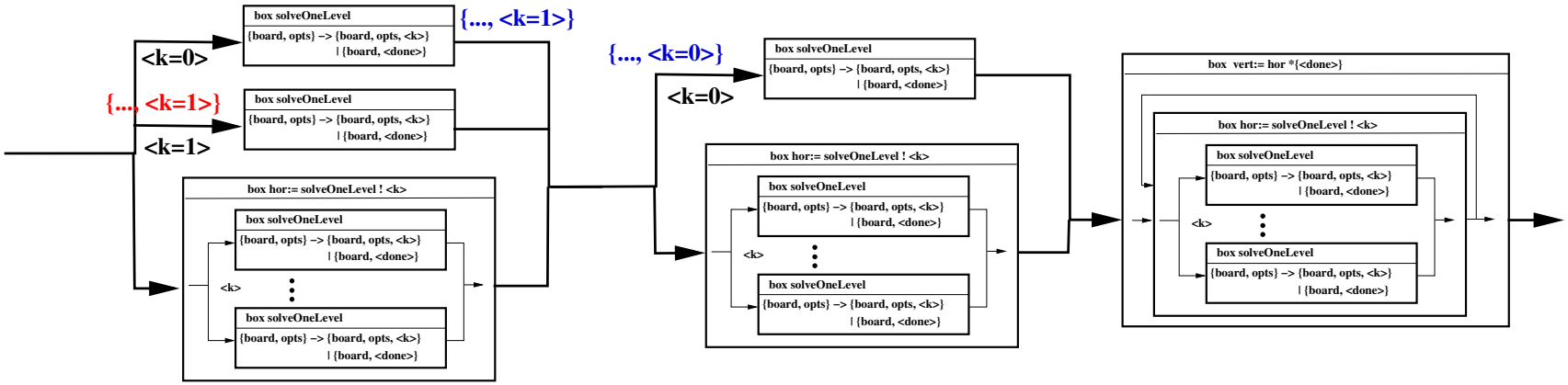
How to apply this to our problem?



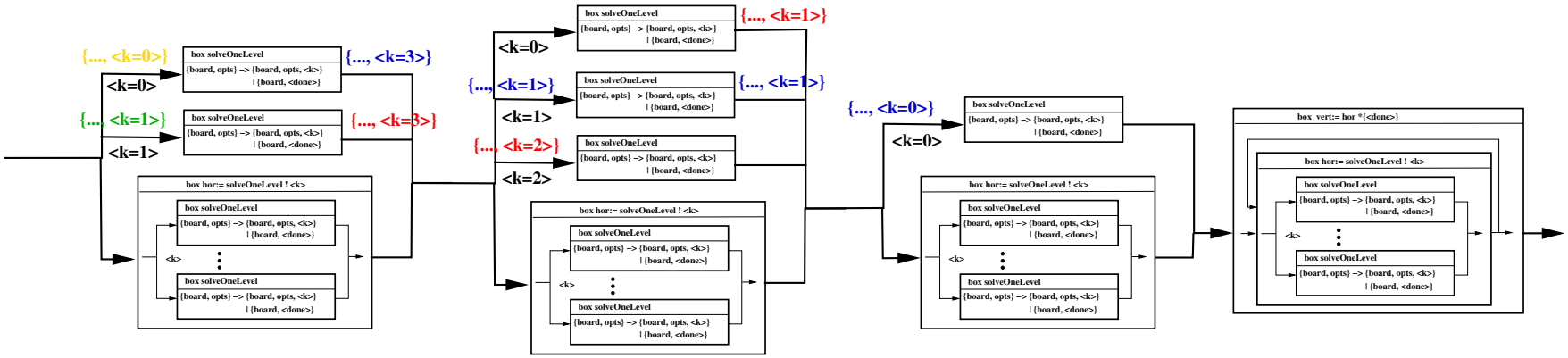
How to apply this to our problem?



How to apply this to our problem?



How to apply this to our problem?



Conclusion

S-Net is

- versatile component framework
- for functional stream processing

S-Net features

- SISO as predominant construction principle
- asynchronous distributed computing
- dynamic network extension
- open box language interface
- few powerfull combinators

Current and Future Work

- prototype implementation (UH + ICL)
  - runtime system
  - compiler
- graphical tools (UH + VTT)
  - network editor
  - network simulator
- mapping to MicroThread architecture (UH + UvA)
  - hardware support for light threads
  - highly efficient synchronisation
- applications (UH + VTT + others)
  - network protocols
  - physics simulations
  - ...