nano-tera.ch
swiss scientific initiative in health / security / environment systems

UltrasoundToGo RTD 2013

FNSNF

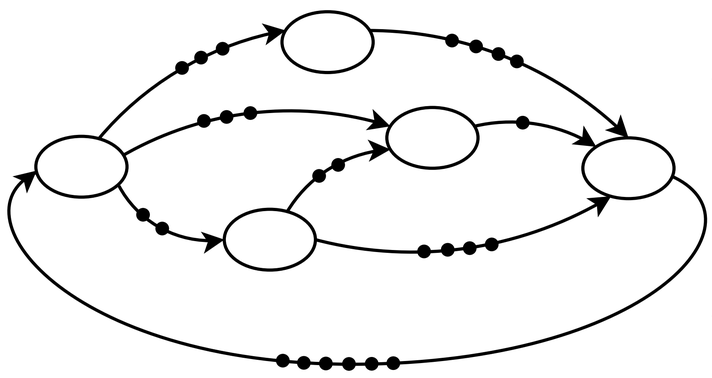
Deterministic Memory Sharing on Kahn Process Networks: Ultrasound Imaging as a Case Study

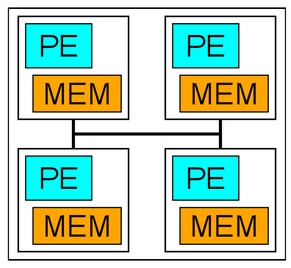
Andreas Tretter, Harshavardhan Pandit, Pratyush Kumar and Lothar Thiele
Computer Engineering and Networks Laboratory (TIK)

TIK ETH zürich

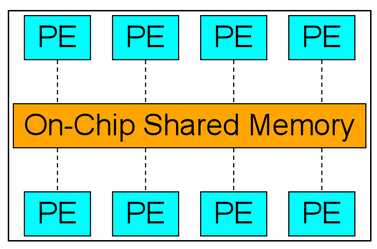
Kahn Process Networks Implemented on Different Platform Types

Kahn Process Networks:
- Popular programming model
- Independent *processes* with separate memory spaces
- Communication by sending *tokens* over *channels*
- Proven to be race-free and scheduling-independent, i.e. **deterministic**





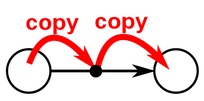
Distributed Memory Systems:
- Cores communicate via message passing
- KPN model fits the hardware well
- Channels can be implemented 1 to 1
- High efficiency



Shared Memory Systems:
- Cores communicate over memory
- Dissimilar to KPN
- Channels are software-emulated
- Efficient native communication methods cannot be exploited

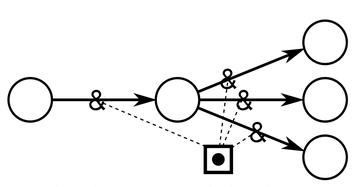
Memory Sharing in Kahn Process Networks

Sending Pointers over Channels

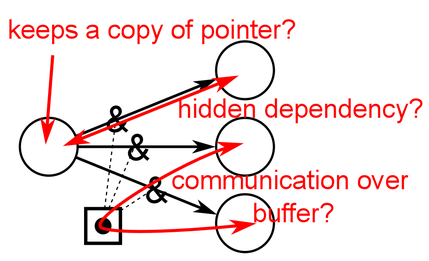


Classic Channel Implementation: Every token deep-copied twice on each channel

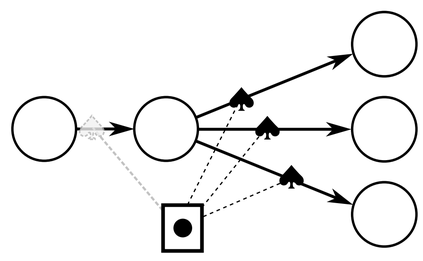
Sending pointers:
- All Processes can access the data in the same position
- Only pointers are copied
- Optimised native data transport methods are exploited



Problems of Pointers



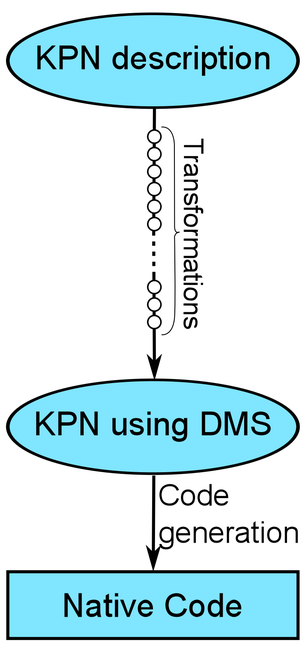
Solution Concept: Using Access Tokens



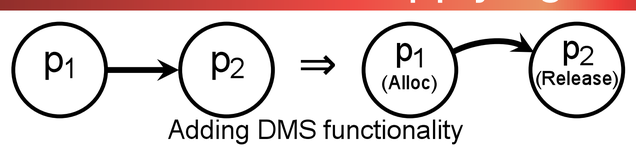
Access tokens allow processes to access a buffer
- Sending them is destructive
- No “wild” duplication
- Only read access if there are multiple access tokens

⇒ KPN properties are preserved
⇒ Application is determinate

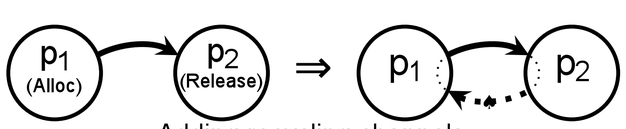
Applying Deterministic Memory Sharing Techniques to Kahn Process Networks



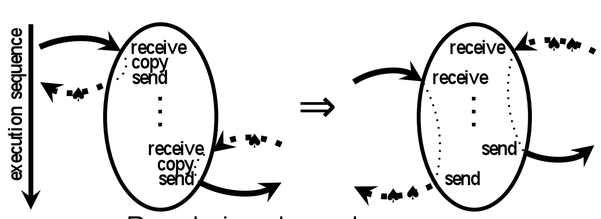
Transformations 1: Applying DMS



Adding DMS functionality

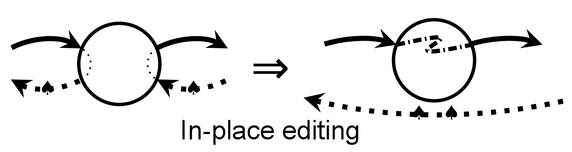


Adding recycling channels
reuse memory blocks instead of dynamic (de)allocation

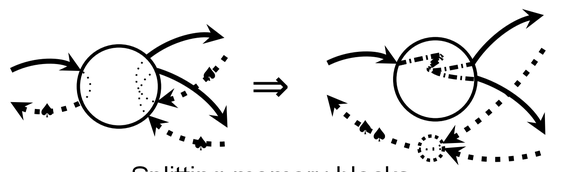


Reordering channel accesses
to access multiple blocks simultaneously

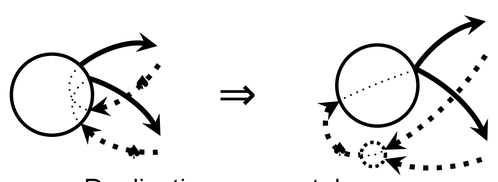
Transforms 2: Optimisation



In-place editing

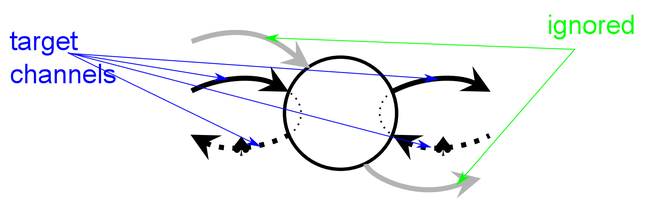


Splitting memory blocks



Duplicating access tokens

About the Transformations



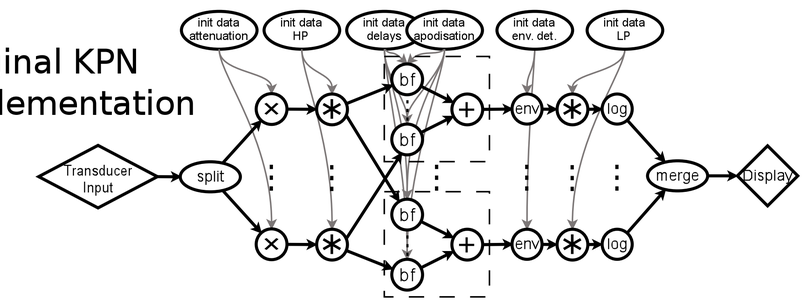
- Looking for certain patterns
- Applicable only under certain conditions
- Transform one process/channel at a time
- Make annotations to channels to ensure integrity of the process network

- Each transformation preserves the application semantics
- Determinacy is never affected

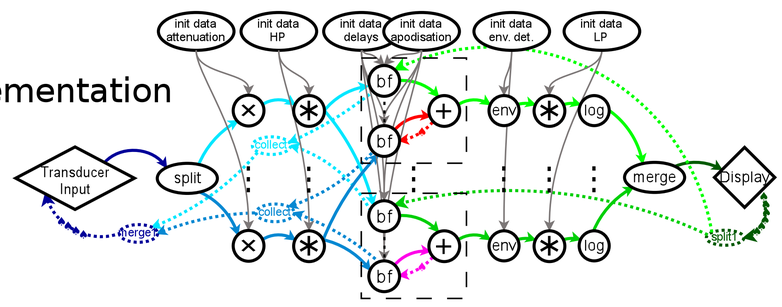
⇒ Correct by construction

Case Study: Medical Ultrasound Imaging

Original KPN Implementation



DMS Implementation



Experimental Evaluation:
On Intel Xeon Phi 5110P accelerator
- 60 cores @ 1053 MHz
- 4 instruction pipelines/core
- Communication over memory/ cache synchronisation

Different configurations tested
- 4000 threads (left), 200 threads
- Dynamic and static mapping
- Traditional channels vs. DMS vs. windowed FIFOs
- Different channel sizes (best performance and good performance/ memory tradeoff shown here)