Formal Abstractions for Packet Scheduling

Mohan, Liu, Foster, Kappé, Kozen
SDN made networks programmable.
SDN made networks programmable.

Early goal: routing.
SDN made networks programmable.

Early goal: routing.

But now we need control over scheduling.
SDN made networks programmable.

Early goal: routing.

But now we need control over *scheduling*.
SDN made networks programmable.

Early goal: routing.

But now we need control over scheduling.

Basic tools work fine...
SDN made networks programmable.

Early goal: routing.

But now we need control over scheduling.

Basic tools work fine…
SDN made networks programmable.

Early goal: routing.

But now we need control over scheduling.

Basic tools work fine…
But modern scheduling requires more.
But modern scheduling requires more.

Traffic goes to either Porto or Torres Vedras.
But modern scheduling requires more.

R traffic goes to either Porto or Torres Vedras.

Goal:

Interleave R and B; interleave P and T.
But modern scheduling requires more.

Traffic goes to either Porto or Torres Vedras.

Goal:
But modern scheduling requires more.

Traffic goes to either Porto or Torres Vedras.

Goal:
New plan!
New plan!
Interleave small, medium, and large packets.
New plan!
Interleave small, medium, and large packets.
New plan! Interleave small, medium, and large packets.
No general way to deploy our gadget.
No general way to deploy our gadget.

A human needs a range of trees.
No general way to deploy our gadget.

A human needs a *range* of trees.

The hardware wants to support *one* tree.
No general way to deploy our gadget.

A human needs a range of trees.

The hardware wants to support one tree.
No general way to deploy our gadget.

A human needs a *range* of trees.

The hardware wants to support *one* tree.
No general way to deploy our gadget.

A human needs a range of trees.

The hardware wants to support one tree.

this work
No general way to deploy our gadget.

A human needs a range of trees.

The hardware wants to support one tree.
Aside: PIFO trees

interleave R and B; interleave P and T.
Aside: PIFO trees

interleave $R$ and $B$; interleave $P$ and $T$. 
Aside: PIFO trees

interleave R and B; interleave P and T.
Aside: PIFO trees

This behaves like a queue!

interleave R and B; interleave P and T.
Aside: PIFO trees

This behaves like a queue!

How do we pop it?

interleave R and B; interleave P and T.
Aside: PIFO trees

interleave R and B; interleave P and T.

This behaves like a queue!

How do we pop it?
Aside: PIFO trees

interleave R and B; interleave P and T.

This behaves like a queue!
How do we pop it?
Aside: PIFO trees

interleave R and B; interleave P and T.

This behaves like a queue!

How do we pop it?

2, 2, 1, 2

1

2

P₂, P₁

B₃, B₂, B₁
Aside: PIFO trees

interleave R and B; interleave P and T.

This behaves like a queue!
How do we pop it?
Aside: PIFO trees

interleave R and B; interleave P and T.

This behaves like a queue!
How do we pop it?
Aside: PIFO trees

interleave $R$ and $B$; interleave $P$ and $T$.

This behaves like a queue!

How do we pop it?
Aside: PIFO trees

This behaves like a queue!

How do we pop it?

interleave R and B; interleave P and T.
Aside: PIFO trees

interleave R and B; interleave P and T.

This behaves like a queue!
How do we pop it?

$P_2$, $B_3$, $B_2$, $B_1$, $P_1$
Aside: PIFO trees

interleave R and B; interleave P and T.

This behaves like a queue!

How do we pop it?

B₃, B₂, P₂, B₁, P₁
Aside: PIFO trees

This behaves like a queue!

How do we pop it?

interleave R and B; interleave P and T.
Aside: PIFO trees

interleave R and B; interleave P and T.

This behaves like a queue!

How do we pop it?
Aside: PIFO trees

Interleave R and B; interleave P and T.

This behaves like a queue!

How do we pop it?
How do we push into it?
Aside: PIFO trees

interleave R and B; interleave P and T.

push $T_1$
Aside: PIFO trees

interleave R and B; interleave P and T.

push $T_1$
Aside: PIFO trees

interleave R and B; interleave P and T.

push $T_1$
Aside: PIFO trees

interleave R and B; interleave P and T.

push $T_1$
Aside: PIFO trees

interleave R and B; interleave P and T.

push $T_1$
Aside: PIFO trees

interleave R and B; interleave P and T.

push $T_1$
Aside: PIFO trees

interleave R and B; interleave P and T.

push $T_1$
Aside: PIFO trees

interleave $\mathbf{R}$ and $\mathbf{B}$; interleave $\mathbf{P}$ and $\mathbf{T}$.

push $T_1$

$2, 2, 1, 2, 1$

$2, 1, 2, 1, 2, 1$

$\mathbf{P}_2, \mathbf{P}_1$

$\mathbf{B}_3, \mathbf{B}_2, \mathbf{B}_1$

$\mathbf{P}_2, \mathbf{T}_1, \mathbf{P}_1$

$\mathbf{B}_3, \mathbf{B}_2, \mathbf{B}_1$

$\mathbf{B}_3, \mathbf{P}_2, \mathbf{B}_2, \mathbf{T}_1, \mathbf{B}_1, \mathbf{P}_1$
Aside: PIFO trees

interleave R and B; interleave P and T.

push $T_1$
Aside: PIFO trees

interleave R and B; interleave P and T.

push $T_1$
Key Insight
Key Insight

A PIFO tree manifests a *programming language.*
Key Insight

A PIFO tree manifests a *programming language*.

A program is precisely a *scheduling algorithm*.
Key Insight

A PIFO tree manifests a programming language.

A program is precisely a scheduling algorithm.
Key Insight

A PIFO tree manifests a programming language.

A program is precisely a scheduling algorithm.

Path: \([(2, r_1), (B_1, r_2)]\)
Key Insight

A PIFO tree manifests a programming language.

A program is precisely a scheduling algorithm.

Path: $[(2,r_1),(B_1,r_2)]$
Key Insight

A PIFO tree manifests a *programming language*.

A program is precisely a *scheduling algorithm*.
Key Insight

A PIFO tree manifests a programming language.

A program is precisely a scheduling algorithm.

tree shape  language expressivity
Which leads to some very PL-ey questions:

tree  language
shape  expressivity
Which leads to some very PL-ey questions:

Tree shape ← language expressivity

Compare expressivity of languages?
Which leads to some very PL-ey questions:

Compare expressivity of languages?
Compare expressivity of trees?
Which leads to some very PL-ey questions:

- Tree shape
- Language expressivity

Compare expressivity of languages?
Compare expressivity of trees?

Compile a program so it runs against a new tree?
A human needs a range of trees.

The hardware wants to support one tree.

No general way to deploy our gadget.
No general way to deploy our gadget.

A human needs a range of trees.

The hardware wants to support one tree.
A human needs a range of trees.

The hardware wants to support one tree.

No general way to deploy our gadget.

compilation

some sufficiently expressive tree
Contributions
Contributions

Formal model of PIFO trees
Contributions

Formal model of PIFO trees

General theorems of expressiveness
w.r.t. tree shape
Contributions

Formal model of PIFO trees
General theorems of expressiveness w.r.t. tree shape
Compiler
Contributions

Formal model of PIFO trees

General theorems of expressiveness w.r.t. tree shape

Compiler

Simulator
Expressivity of trees

Trees with more leaves are more expressive. Taller trees are more expressive.
Expressivity of trees

Trees with more leaves are more expressive. Taller trees are more expressive.

Captured elegantly by:

*Homomorphic embedding.* Map root to root, leaves to leaves. Respect ancestry.
Expressivity of trees

Homomorphic embedding.
Map root to root, leaves to leaves. Respect ancestry.
Expressivity of trees

Homomorphic embedding.
Map root to root, leaves to leaves. Respect ancestry.
Expressivity of trees

Homomorphomic embedding.
Map root to root, leaves to leaves. Respect ancestry.
Compiling programs
Compiling programs

Diagram 1:

Diagram 2:
Compiling programs

Diagram showing the process of compiling programs with nodes and arrows indicating the flow.
Compiling programs
Compiling programs
Compiling programs
Compiling programs
Compiling programs
Compiling programs
Compiling programs

1, 3, 1, 2, 3, 1, 2, 3

\[ \ldots \]

\[ \ldots \]

\[ \ldots \]
Compiling programs

1, 3, 1, 2, 3, 1, 2, 3

1, 3, 1, 2, 3, 1, 2, 3
Compiling programs
Compiling programs

1, 3, 1, 2, 3, 1, 2, 3

1, 3, 1, 2, 3, 1, 2, 3

1, 2, 1, 2, 2, 1, 2, 2

3, 2, 3, 2, 3

1

2

3, 2, 3, 2, 3

1

2

…

…

…

…

…
Compiling programs

1, 3, 1, 2, 3, 1, 2, 3

1, 3, 1, 2, 3

1, 2, 1, 2, 2, 1, 2, 2
Compiling programs

1, 3, 1, 2, 3, 1, 2, 3

1, 3, 1, 2, 3, 1, 2, 3

1, 2, 1, 2, 2, 1, 2, 2

1, 2, 1, 2, 2, 1, 2, 2
Compiling programs

Path: $[(2, r_1), \ldots]$
Compiling programs

Path: \([(2, r_1), \ldots]\)

1, 2, 3, 1, 2, 3, 1, 2, 3

B

1, 2, 1, 2, 2, 1, 2, 2
Compiling programs

Path: \[(1, r_1), \ldots\]

Path: \[(2, ?), (1, ?), \ldots\]
Compiling programs

Path: \[(2, r_1), \ldots\]

1, 2, 3, 1, 2, 3, 1, 2, 3

Path: \[(1, r_1), \ldots\]

1, 2, 1, 2, 2, 1, 2, 2

B

Path: \[(2), \ldots\]

...
Compiling programs

Path: [(2, r₁), ...]

Path: [(1, r₁), ...]
Compiling programs

Path: [(2, r_1), ...]

Path: [(1, r_1), ...]
Given an embedding, we *lift* it to arrive at a compiler.
Generating embeddings automatically!
Generating embeddings automatically!

**Homomorphic embedding.**
Map root to root, leaves to leaves. Respect ancestry.
Generating embeddings automatically!

*Homomorphic embedding.*
Map root to root, leaves to leaves. Respect ancestry.

Two new algorithms, both starting with heterogeneous source trees.
Generating embeddings automatically!

Homomorphic embedding.
Map root to root, leaves to leaves. Respect ancestry.

Two new algorithms,
both starting with heterogeneous source trees.

1. If target tree is regular $d$-ary for some $d$. 
Generating embeddings automatically!

*Homomorphic embedding.*
Map root to root, leaves to leaves. Respect ancestry.

Two new algorithms, both starting with heterogeneous source trees.

1. If target tree is regular $d$-ary for some $d$.
2. If target tree is itself heterogeneous.
Workflow

WFQ: 40/40/20

A → B → RR

C → D → WFQ: 10/40/50

E → F → G
Workflow

logical
But the hardware supports a regular-branching binary tree.
But the hardware supports a regular-branching binary tree.

No problem.
Here’s how I’ll use that tree.
No problem.  
Here’s how I’ll use that tree.
Workflow

WFQ: 40/40/20
A → B → RR
C → D → WFQ: 10/40/50
E → F → G

logical

WFQ: 40/40/20
T → RR
A → B → T → WFQ: 10/40/50
C → D → T → G
E → F

actual
Simulation

logical

actual

WFQ: 40/40/20
A → B → RR

WFQ: 10/40/50
C → D

WFQ: 40/40/20
T → RR

WFQ: 10/40/50
C → D → T → G

Simulation diagram with logical and actual flows, showing WFQ weights and node connections.
Simulation

logical

actual
Underlying formalism
Underlying formalism

\[
\begin{align*}
\text{Path} & \quad \text{PIFOTree} \\
* \in \text{Topo} & \quad n \in \mathbb{N} \quad ts \in \text{Topo}^n \\
\text{Node}(ts) \in \text{Topo} & \\
\end{align*}
\]

\[
\begin{align*}
p \in \text{PIFO(Pkt)} & \quad \forall 1 \leq i \leq n. \, qs[i] \in \text{PIFOTree}(ts[i]) \\
\text{Leaf}(p) \in \text{PIFOTree}(*) & \quad \text{Internal}(qs, p) \in \text{PIFOTree}(\text{Node}(ts)) \\
\text{Topo} & \quad \text{Topo} \\
\end{align*}
\]

\[
\begin{align*}
r \in \text{Rk} & \quad ts \in \text{Topo}^n \quad 1 \leq i \leq n \quad r \in \text{Rk} \quad pt \in \text{Path}(ts[i]) \\
r \in \text{Path}(*) & \quad (i, r) :: pt \in \text{Path}(\text{Node}(ts)) \\
\text{Topo} & \quad \text{Topo} \\
\end{align*}
\]

\[
\begin{align*}
push(p, pkt, r) = p' & \quad \text{push}(qs[i], pkt, pt) = q' \\
push(\text{Leaf}(p), pkt, r) = \text{Leaf}(p') & \quad \text{PUSH}(p, i, r) = p' \\
push(\text{Internal}(qs, p), pkt, (i, r) :: pt) = \text{Internal}(qs[i/q'], p') & \\
\end{align*}
\]

\[
\begin{align*}
push(\text{Leaf}(p), pkt, r) = \text{Leaf}(p') & \quad \text{PUSH}(p, i, r) = p' \\
push(\text{Internal}(qs, p), pkt, (i, r) :: pt) = \text{Internal}(qs[i/q'], p') & \\
\end{align*}
\]
A general way to deploy PIFO trees.
A general way to deploy PIFO trees.

Let the hardware support some tree.
A general way to deploy PIFO trees.

Let the hardware support some tree.

Let the human program against some tree.
A general way to deploy PIFO trees.

Let the human program against some tree.

Let the hardware support some tree.
A general way to deploy PIFO trees.

Let the human program against some tree.

Let the hardware support some tree.

Compilable?
A general way to deploy PIFO trees.

Let the human program against some tree.

Let the hardware support some tree.
A general way to deploy PIFO trees.

Let the human program against some tree.

Let the hardware support some tree.

Compilable?
A general way to deploy PIFO trees.

Let the human program against some tree.

Let the hardware support some tree.
A general way to deploy PIFO trees.

Let the human program against some tree.

Let the hardware support some tree.

Compilable?
A general way to deploy PIFO trees.

Let the human program against some tree.

Let the hardware support some tree.

Compilable?
A general way to deploy PIFO trees.

Let the human program against some tree.

Let the hardware support some tree.

Compilable?
A general way to deploy PIFO trees.

Let the human program against some tree.

Let the hardware support some tree.
Let the hardware support some tree.

Compilable?

A general way to deploy PIFO trees.

Let the human program against some tree.

Let the hardware support some tree.
A general way to deploy PIFO trees.

Let the human program against some tree.

Compilable?

Let the hardware support some tree.
Formal Abstractions for Packet Scheduling

Mohan, Liu, Foster, Kappé, Kozen

cs.cornell.edu/~amohan
Formal Abstractions for Packet Scheduling

Mohan, Liu, Foster, Kappé, Kozen

cs.cornell.edu/~amohan