Summer School on Software Synthesis

Schloss Dagstuhl

August 8-12 2011

Ras Bodik, Sumit Gulwani, Viktor Kuncak, Eran Yahav

Welcome to Schloss Dagstuhl

A “computer science monastery” with its own ghost

Seclusion facilitating communication

Logistics of your stay: talk to me or the reception

- get familiar with facilities (bikes, table tennis, billiard)
- think of a trip for Wed afternoon
- pay when checking out

History of this summer school

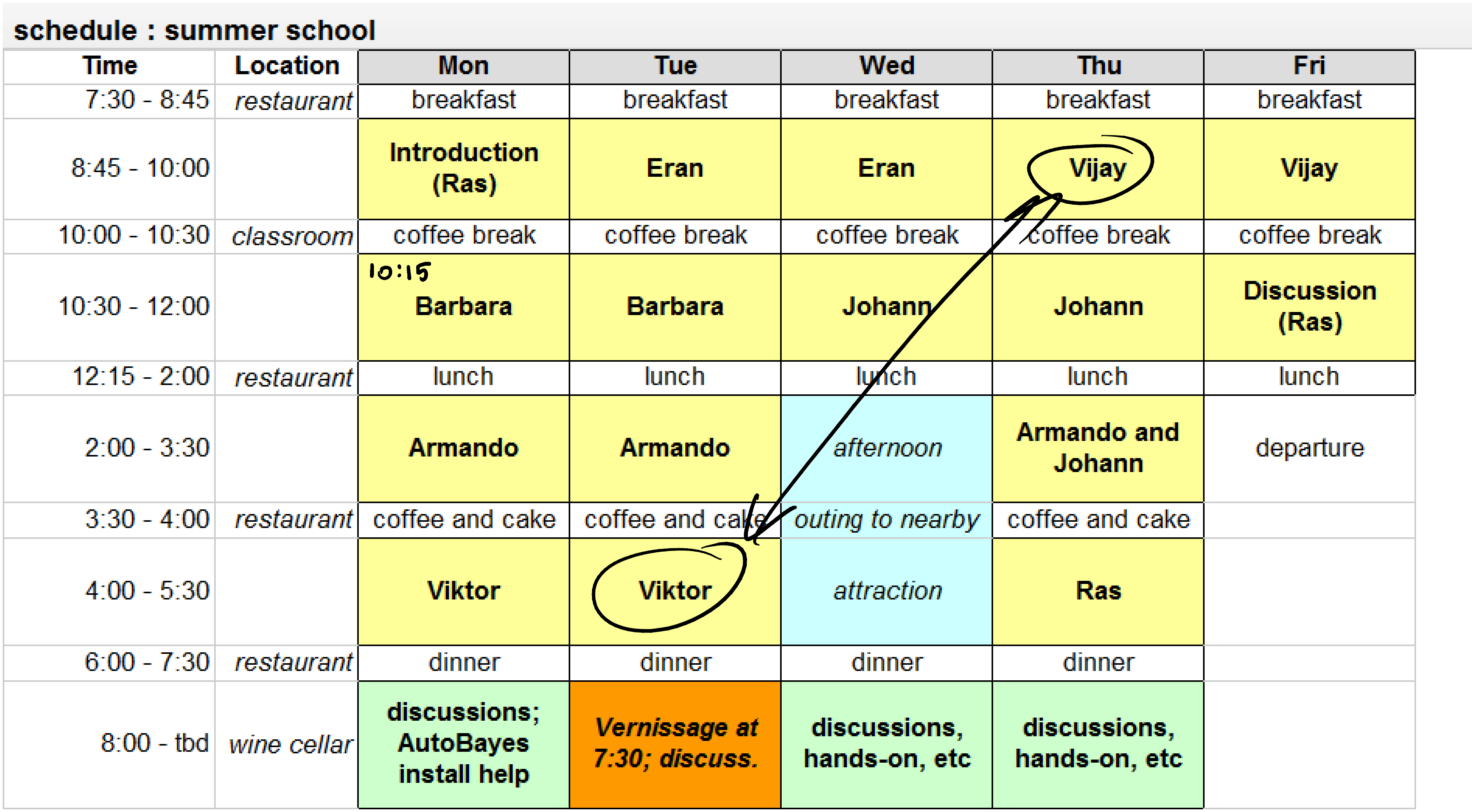
Dagstuhl Seminar on Software Synthesis, Dec 2009

- brought together several communities
- they gave tutorials to each other
- very well received seminar ==> what to do next?

Idea: give these tutorial talks to PhD students

- after all, they do all the work
- and hence will advance the field
- again, a spectrum of approaches to be presented

Schedule



schedule : summer school

Time	Location	Mon	Tue	Wed	Thu	Fri
7:30 - 8:45	*restaurant*	breakfast	breakfast	breakfast	breakfast	breakfast
8:45 - 10:00		**Introduction (Ras)**	**Eran**	**Eran**	**Vijay**	**Vijay**
10:00 - 10:30	*classroom*	coffee break	coffee break	coffee break	coffee break	coffee break
10:30 - 12:00		10:15 **Barbara**	**Barbara**	**Johann**	**Johann**	**Discussion (Ras)**
12:15 - 2:00	*restaurant*	lunch	lunch	lunch	lunch	lunch
2:00 - 3:30		**Armando**	**Armando**	*afternoon*	**Armando and Johann**	departure
3:30 - 4:00	*restaurant*	coffee and cake	coffee and cake	*outing to nearby*	coffee and cake	
4:00 - 5:30		**Viktor**	**Viktor**	*attraction*	**Ras**	
6:00 - 7:30	*restaurant*	dinner	dinner	dinner	dinner	
8:00 - tbd	*wine cellar*	**discussions; AutoBayes install help**	***Vernissage at 7:30; discuss.***	**discussions, hands-on, etc**	**discussions, hands-on, etc**	

Introductions

Name

school

research interests

thesis topic or still looking?

Hobbies:

what you’d be doing this week if you were not here?

Why are we here?

- design from src
- model from traces

Synthesis 2.0:

- exec. monitors
- reverse engineering

Resurgence of synthesis in several communities

- embedded comp. Matlab → executable
- HPC, clusters of multicores
- HW synth C → RTL

autotuning

HCI - end-user programs / from demos
- query optimization in DB
- robotics
- compiler optimization
  - numerical SW

- test case gen
- spec. synth.
- spec → mockup

Why are we here?

Synthesis 2.0:

Resurgence of synthesis in several communities

- *Deductive Synthesis*
- *Synthesis from demonstration*
- *Controller synthesis*
- *Transformational synthesis with performance exploration*
- *Synthesis of loop invariants for verification*
- *Partial programs for intelligent agent programming*
- *Bug repair*
- *Efficient program space exploration*

What is synthesis?

Wikipedia:

interactive

***Program synthesis** comprises a range of technologies for the automatic generation of executable computer programs from high-level specifications of their behaviour. In contrast to compilation, the specifications are usually non-algorithmic.*

executable

I/O + structural

also backwards

What is synthesis?

is synthesis

PBD

compiler that does not work

a rewrite-based compiler

is not synthesis

----

manual coding

What is synthesis?

is synthesis

----

search / constraint solving

semantics-aware

resolves freedom in specification

choose order of transformation

specs are declarative

is not synthesis

----

optimizing compiler (deterministic xformation)

syntax-directed

Successes

- Academic
  - QuickCode (PBD for Excel)
  - SmartEdit
  - SpecWare
- Industrial
  - Autobayes
  - Autofilter
  - FFTW, Spiral
  - HW synth
  - UML → code

Academic successes

- Constable: programs from constructive proofs
- Play-In-Play-Out
- SmartEdit: programming by demonstration

- genetic programming
- SKETCHING
- DSLs (SQL, LabView, Matlab ?)
- super optimizers
- inductive synthesis of regular languages

Industrial successes

- KIDS
- FFTW, SPIRAL
- autotuning (linear algebra libraries)
- program refinement (B method)
- hw synthesis, incl. $C \rightarrow$ FPGA
- controller synthesis

Why synthesis now?

Needs (challenge problems)

multicore pgming

end-user pgming

safety critical

Why synthesis now?

Opportunities (why became possible to do now)

cycles, not smarts

- search, constraint solving

algorithms

abstractions / DSL / libs

funding available

lessons of verification failures

derive from design,
don't verify

Some themes

- How to obtain a specification?
- How to develop and debug domain theories?
- How to exploit recent advances in verification and decision procedures?
- What lessons can be learnt from success/failure stories?
- What problems could lead to great dissertations?

Another round of introductions

- What is synthesis?
- What drew you to synthesis?
- What artifacts do you want to synthesize?
- What is the input to your "synthesizer"?
- What techniques do you use?
- A modest success:
- A spectacular failure:
- In Dagstuhl, I want to understand X

Logistics

After dinner cheese and wine in the cellar

Wed afternoon / evening outing

What would you like from this school