INFORMS Annual meeting 2016
Nashville, TN

AMPL representation and solution of multiple stochastic programming formulations

Christian Valente, Victor Zverovich, Gautam Mitra,
Robert Fourer

OptiRisk SYSTEMS

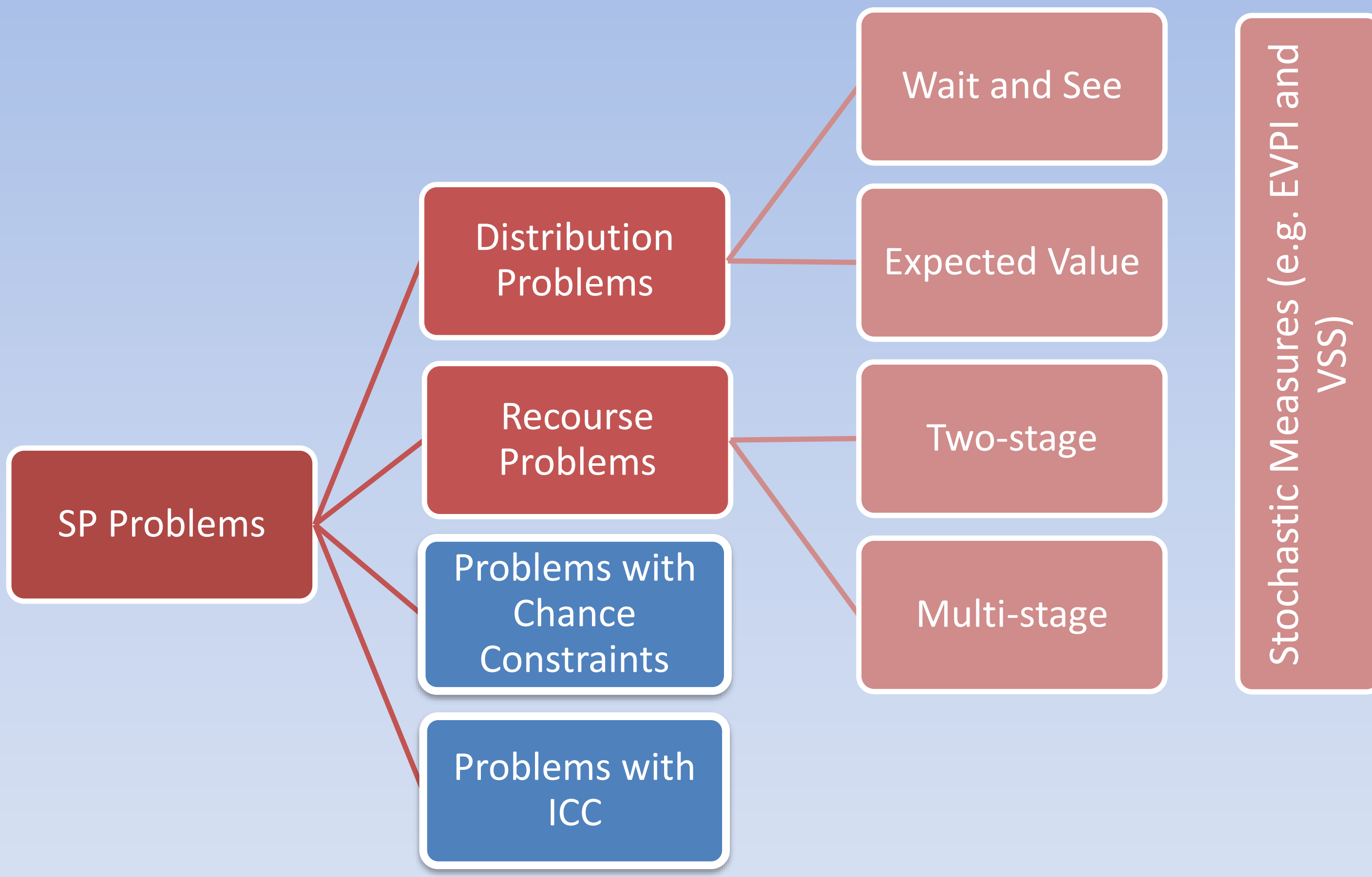
Agenda

- Classification of problems of interest
- Our approach to modelling
- AMPLsp
- Conclusions

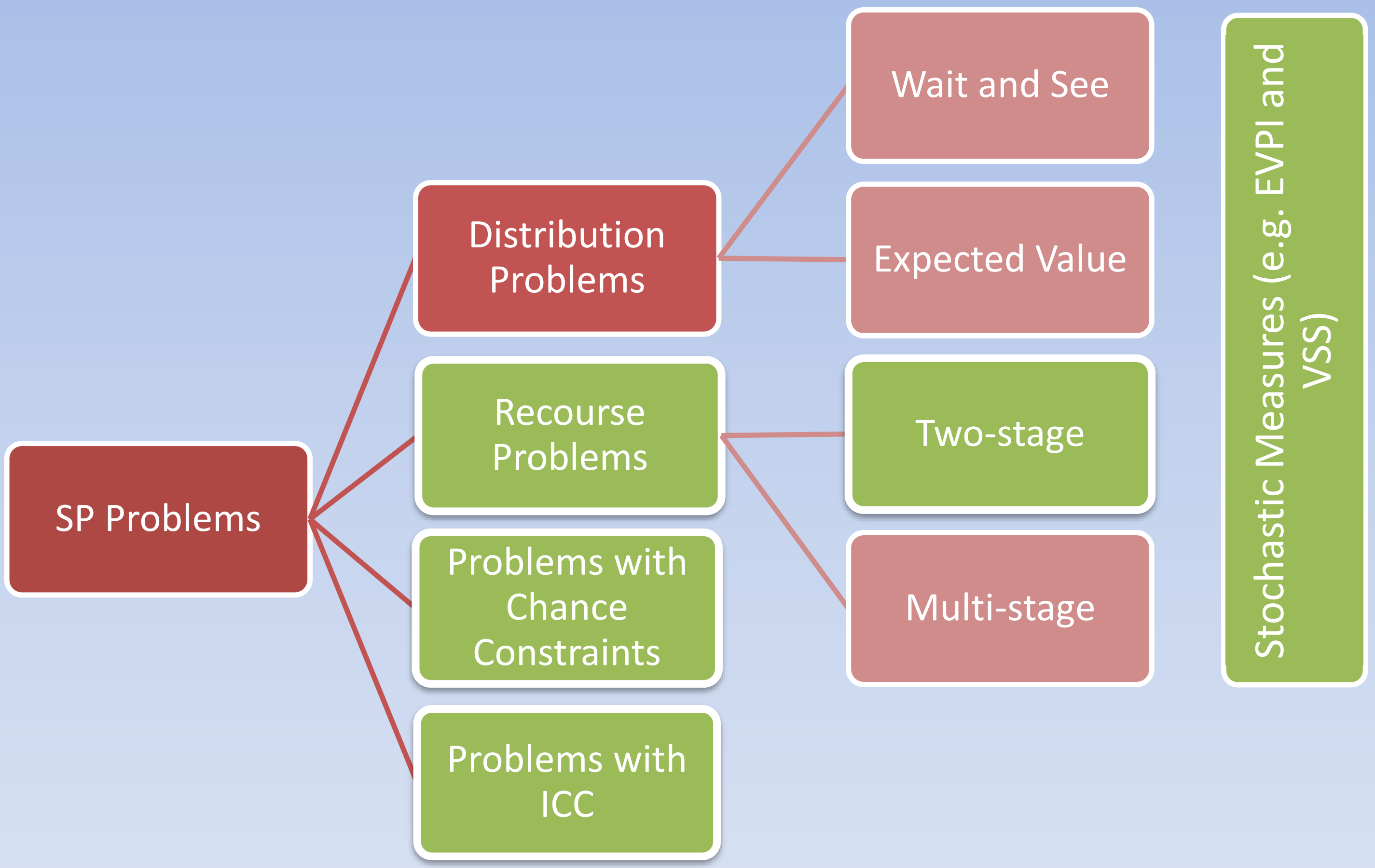
About us..

- In the past years, **OptiRisk Systems** has been working closely with AMPL Inc. and has developed various products in the AMPL ecosystem
  - AMPL Studio (graphical interface)
  - AMPLCOM (library)
  - **SPInE and SAMPL (extensions to AMPL)**
  - FortSP (decomposition based solver)
  - AMPLDev (graphical interface)
  - AMPL API and AMPL IDE (as contractors)

Taxonomy of optimisation problems under uncertainty revised



Taxonomy of optimisation problems under uncertainty revised



- We concentrate on Two-Stage SP problems, with (integrated) chance constraints

Model classes

- Expected Value

$$Z_{EV} \stackrel{\text{def}}{=} \min \bar{c}x$$

where $x \in \bar{F}$

and $\bar{F} = \{x | \bar{A}x = \bar{b}, x \geq 0\}$

- Wait and See

$$Z_{WS} \stackrel{\text{def}}{=} E_{\omega}\left[Z^{\omega}\right]$$

where $Z^{\omega} = \min c^{\omega}x$

and $x \in F^{\omega}$

- Here and now

$$Z_{HN} \stackrel{\text{def}}{=} \min cx + E_{\omega}[\, Q(x,\omega)]$$

subject to $Ax = b$

$x \geq 0$

where $Q(x,\omega) = \min f^{\omega} y^{\omega}$

$D^{\omega} y^{\omega} = d^{\omega} + B^{\omega} x$

$y^{\omega} \geq 0$

$$Z_{HN} \stackrel{\text{def}}{=} \min E_{\omega}\left[c^{\omega}x\right]$$

where $x \in F$ and $F = \bigcap_{\omega \Omega \in} F^{\omega}$

Chance Constraints

- Algebraic formulation:
  - Individual Chance Constraints

$$P(h_i(x,\xi) \geq 0) \geq p_i, \qquad i \in I$$

  - Joint Chance Constraints

$$P((h_i(x,\xi) \geq 0, i \in I) \geq p$$

where *x* and ξ are respectively decisions and random vectors, I is a set of indices of constraints in the given problem

Chance Constraints

- Practical Importance
  - Chance constraints provide a simple risk measure
  - Related to VaR
  - Applications in finance, energy production, water management, …
- Can be expressed in any AMLs reformulating the problem by introducing extra constraints and binary variables

Integrated Chance Constraints

- Expected violation of constraint(s) <= shortfall β_i
- Individual ICC

$$E_\omega[\eta_i(x,\omega)^-] \leq \beta_i, \qquad \beta_i \geq 0, i \in I$$

- Joint ICC

$$E_\omega[\max_{i\in I} \eta_i(x,\omega)^-] \leq \beta_i, \qquad \beta_i \geq 0$$

where $\eta_i(x,\omega)^-$ represents the violation that occurs in constraint *i* under realisation ω

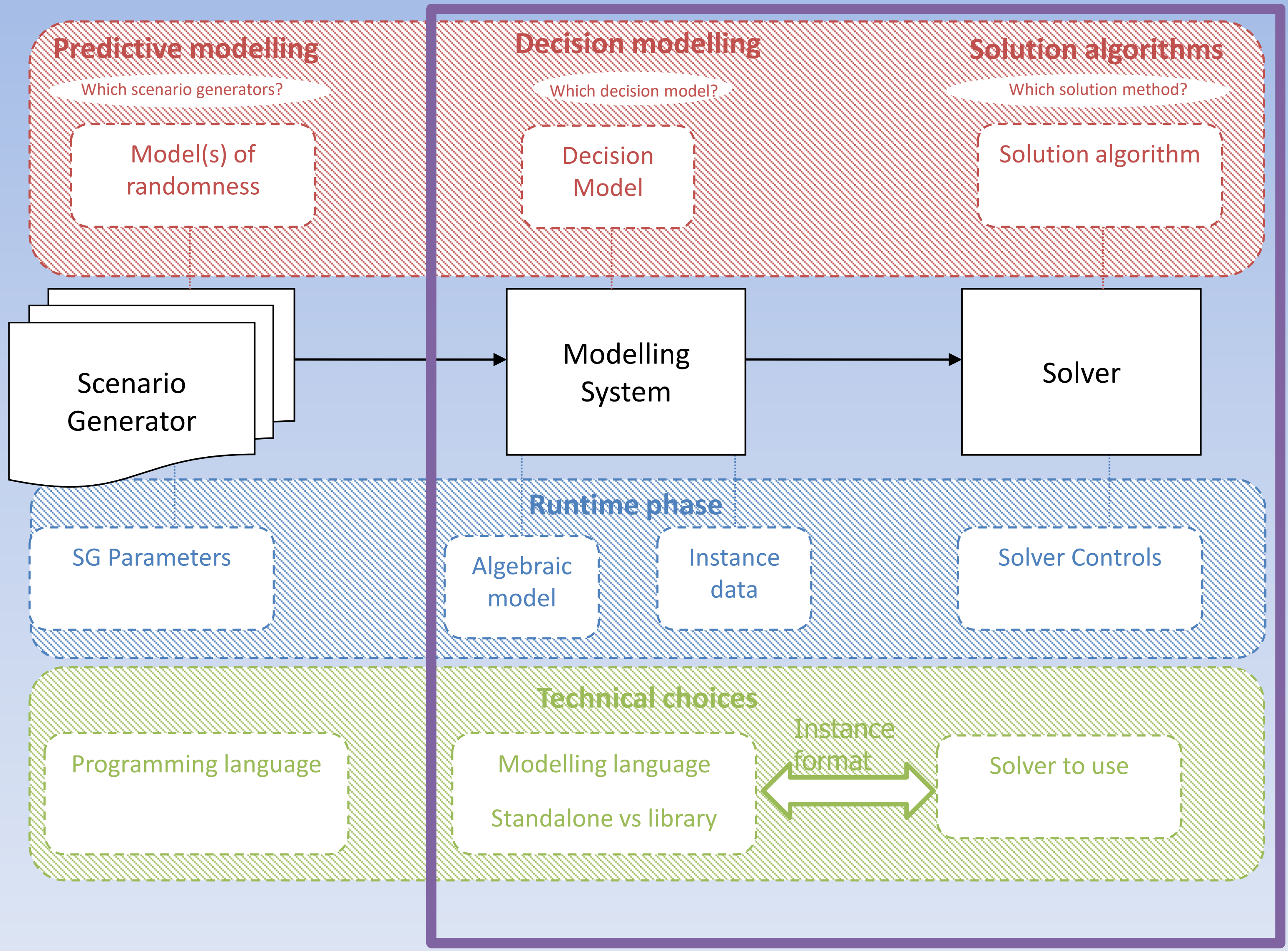
Integrated Chance Constraints

- Practical Importance
  - ICCs represent a risk measure (closely related to CVaR or to SSD)
  - Computationally more tractable than chance constraints
  - Applications in finance, e.g. asset-liability management, portfolio choice models
- Can be expressed in any AML introducing extra constraints and continuous variables

Our approach to optimisation (under uncertainty)

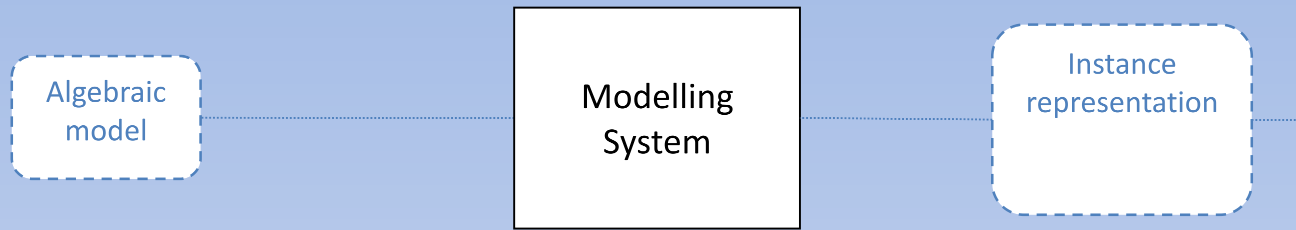
- Maintain separation between the activities in optimisation:
  - Modelling
  - Instance generation
  - Solving
- Benefits
  - Easier specification of the algebraic model
  - Modularity makes software easier to maintain
  - Specialists can work in their own domain

SP Modelling process



OptiRisk SYSTEMS

Our approach to modelling



AML

- How to define the model at algebraic level

AMS

- What modelling system to use

Instance level format

- How to represent the model instance

Introduction
Classification
Our approach to SP
AMPLsp
Conclusions

SP Instance representation

- SP problems have a specific block structure

$$
\begin{aligned}
\min cx \quad & + c_{s1} y_{s1} \quad + c_{s2} y_{s2} \\
Ax \quad & \qquad\qquad\qquad\qquad \leq b \\
A_{s1,1}x + A_{s1,2}Y_{s1} & \qquad\qquad\qquad \leq b_{s1} \\
A_{s2,1}x \quad & \qquad + A_{s2,2}Y_{s2} \leq b_{s2}
\end{aligned}
$$

- When passed to a solver as a deterministic equivalent, this structure is lost

$$
\begin{aligned}
\min \bar{c}z & \\
\bar{A}z & \leq \bar{b}
\end{aligned}
$$

Where $\bar{A}, \bar{b}, \bar{c}$ and z are compositions of the respective vectors/matrices

SP Instance Representation

- The structure can be exploited by solution algorithms
- At instance level we aim to communicate:

$$\begin{aligned} \min\, & cx + \tilde{c}y \\ & Ax \qquad\qquad\;\; \leq b \\ & \widetilde{A_1}x + \widetilde{A_2}y \leq \tilde{b} \end{aligned}$$

$$\begin{aligned} \tilde{c} &= c_{s1}, c_{s2} \\ b &= b_{s1}, b_{s2} \\ \widetilde{A_1} &= A_{S_1,1}, A_{S_2,1} \\ \widetilde{A_2} &= A_{S_1,2}, A_{S_2,2} \end{aligned}$$

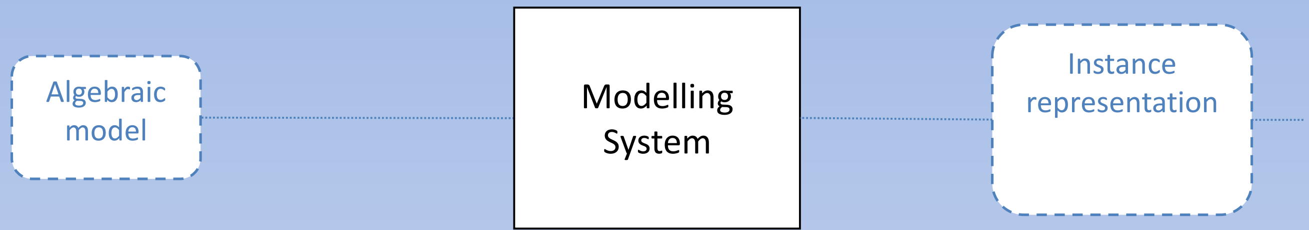
where the tilde submatrices are then separately passed, scenario by scenario.

- Most of the elements in the sub-blocks are repeated -> only changes in respect to the tilde matrices are communicated

SP Instance Representation

- A well specified language for instance level representation has already been proposed and is used (SMPS)
- To be able to generate such format, the modelling system must be told the structure of the model we wish it to convey
- Following slides show our past and current approaches at this

Our approach to modelling (1)



AML

- Any modelling language, no specialized syntax (DEQ formulation)

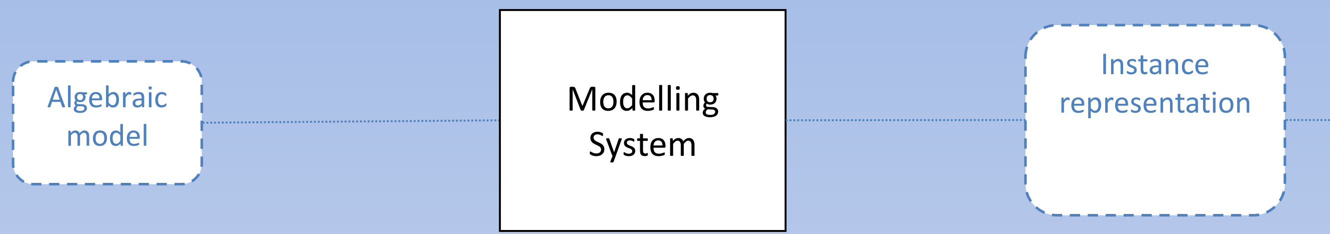
AMS

- Any modelling system for linear/non-linear optimisation

MPS

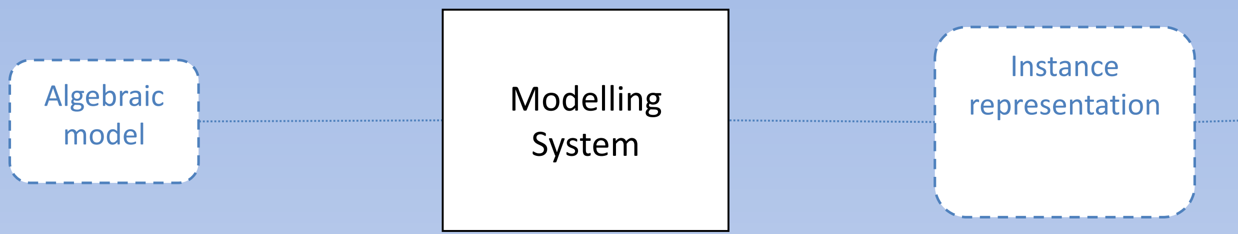
- MPS like format (direct representation of the DEQ formulation)
- Replication of information
- Loss of structure

Our approach to modelling (2)



SAMPL	SPInE	SMPS
• Specialized syntax for SP	• Preprocessor at algebraic language level • Generated a core model and the needed information using AMPL *as a subsystem* • Expressed model != solved model	• SMPS like format (compact representation, conveys the stochastic information separately)

Our approach to modelling (3)



Introduction
Classification
Our approach to SP
AMPLsp
Conclusions

SAMPL	SAMPL	SMPS
• Specialized syntax for SP	• Reimplementation of AMPL • Generated model efficiently • Development independent from AMPL	• SMPS like format (compact representation, conveys the stochastic information separately)

Our approach to modelling (4)

Algebraic model — Modelling System — Instance representation

AMPLsp	AMPL + smpswriter	SMPS
• AMPL formulation following some guidelines	• Official AMPL interpreter • Uses the solver module **smpswriter** to efficiently generate smps	• SMPS like format (compact representation, conveys the stochastic information separately)

OptiRisk SYSTEMS

From SAMPL to AMPLsp

- We have been developing SAMPL, an extended version of AMPL with additional language constructs and models communication facilities
- Focus of the language was:
  - Easy formulation of the classes of problems presented (e.g. no artificial variables for (I)CCP)
  - Efficient model instance generation
  - Efficient model solution: instances generated in SMPS-like format, which conveys the model structure, exploitable by various specialised algorithms

From SAMPL to AMPLsp

- SAMPL has always been separated from AMPL, first implemented as a pre-processor then as an alternative language interpreter
  - Two development teams and efforts
  - Not all AMPL facilities were implemented
  - Sync with new AMPL features
- **SAMPL is discontinued**, to be replaced by
  - AMPL with an intelligent reuse of existing constructs
  - smpswriter [https://github.com/ampl/mp] (a new solver interface, able to write SMPS files)

OptiRisk SYSTEMS

Dakota model (deterministic)


```
set PROD;
set RESOURCE;

param Cost{RESOURCE};
param ProdReq{RESOURCE,PROD};
param Price{PROD};
param Budget;
param Demand{PROD};

var amountbuy{RESOURCE} >=0 ;
var amountprod{PROD}>=0, suffix stage 2;
var amountsell{PROD}>=0, suffix stage 2;

maximize wealth:
      sum{p in PROD} Price[p]*amountsell[p]-
      sum{r in RESOURCE} Cost[r]*amountbuy[r];
```

Dakota model (deterministic)

- Introduction
- Classification
- Our approach to SP
- AMPLsp
- Conclusions

```
subject to

CBudget: sum{r in RESOURCE}
        Cost[r]*amountbuy[r] <= Budget;
CBalance{r in RESOURCE}:
        amountbuy[r] >= sum{p in PROD} ProdReq[r,p] *
        amountprod[p];
CProduction{p in PROD}:
        amountsell[p] <= amountprod[p];
CSales{p in PROD}:
        amountsell[p] <= Demand[p];
```

OptiRisk SYSTEMS

Dakota model (stochastic)

- The implementation is designed to have minimal impact on AMPL by reusing the representational power of the *nl* format and a few conventions
- Preliminary declarations/conventions:

```
function expectation;
function random;
suffix stage IN;
```

- Add scenario set and appropriate indexing to represent realizations:

```
set SCEN;
param Demand{PROD, SCEN};
```

OptiRisk SYSTEMS

Dakota model (stochastic)

- For every occurrence of the random parameter in the model we pass a placeholder
- Parameter **Demand** becomes a variable (in the sense that its value will be determined after AMPL generates the model instance)

```
param Demand{PROD};  →  var RandomDemand{PROD};
```

- An AMPL function allows the smpswriter to link the parameter values to its placeholder

```
yield: random({p in PROD} (Demand[p],
                 {s in SCEN} RandomDemand[p,s]));
```

Dakota model (stochastic)

- Stage assignment

```
var amountbuy{RESOURCE} >=0 ;
var amountprod{PROD}>=0, suffix stage 2;
var amountsell{PROD}>=0, suffix stage 2;
```

- Objective as expectation

```
maximize wealth:
expectation(sum{p in PROD} Price[p]*amountsell[p])
 - sum{r in RESOURCE} Cost[r]*amountbuy[r];
```

OptiRisk SYSTEMS

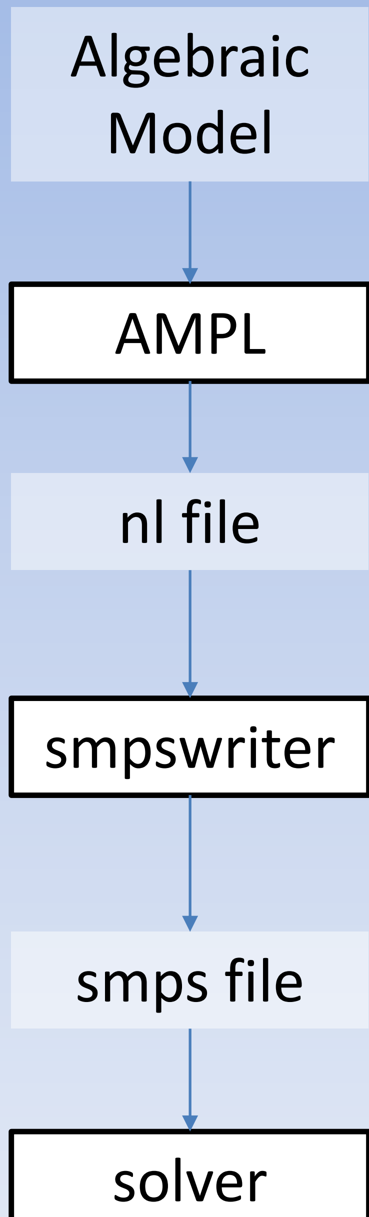
Complete model

```
function expectation;
function random;
suffix stage IN;

set PROD;
set RESOURCE;
set SCEN;
param RandomDemand{PROD, SCEN};
var Demand{PROD};
yield: random({p in PROD} (RandomDemand[p],
 {s in SCEN} Demand[p,s]));
param Cost{RESOURCE};
param ProdReq{RESOURCE,PROD};
param Price{PROD};
param Budget;

var amountbuy{RESOURCE} >=0 ;
var amountprod{PROD}>=0, suffix stage 2;
var amountsell{PROD}>=0, suffix stage 2;
```

Complete model

```
maximize wealth :
expectation(sum{p in PROD} Price[p]*amountsell[p])-
      sum{r in RESOURCE} Cost[r]*amountbuy[r];

subject to
CBudget: sum{r in RESOURCE}
      Cost[r]*amountbuy[r] <= Budget;
CBalance{r in RESOURCE}:
      amountbuy[r] >= sum{p in PROD} ProdReq[r,p] *
amountprod[p];
CProduction{p in PROD}:
      amountsell[p] <= amountprod[p];
CSales{p in PROD}:
      amountsell[p] <= RandomDemand[p];
```

Process




```
write gmodel;
```

```
Reading time = 0.015569 s.
Stage 1 has 1 row(s), 3 column(s), and 3 nonzero(s).
Stage 2 has 9 row(s), 6 column(s), and 21 nonzero(s).
Problem has 2 stage(s) and 3 scenario(s).
Itn      Objective          Bound          Rel.Gap
  1        1281.63        4577.78          2.57184
  2        1281.63           4120          2.21466
  3        1281.63           4120          2.21466
  4        1281.63        2589.71          1.02064
  5        1580.19        2446.55          0.54826
  6        1580.19        2266.71         0.434451
  7        1580.19        1790.67         0.133201
  8        1580.19        1711.66        0.0831986
  9        1580.19        1655.78        0.0478343
 10        1580.19        1616.67        0.0230834
 11        1616.67        1616.67    -2.81287e-016
Number of iterations = 11.
Master time = 0 s.
Recourse time = 0 s.
Optimal solution found, objective = 1616.67.
```

```
shell 'smpswriter model';
```

```
shell 'fortsp model';
```

OptiRisk SYSTEMS

Availability

- Source code is available on github as part of the project ampl/*mp:* An open-source library for mathematical programming

  [*https://github.com/ampl/mp*](https://github.com/ampl/mp)

- Find details in **solvers/smpswriter**

Further developments

- Syntax
  - Streamline the initial declarations
  - Improve the procedure to define a random parameter
- Functionalities
  - Incoroprate CPPs and ICCPs
- Integration
  - Implement the smpswriter as a standard AMPL solver (no need to system calls)
- Variables
  - Allow second stage variables to be indexed over the scenario set

OptiRisk SYSTEMS

Conclusions

- Formulating SP problems with DEQ does not scale up
  - Spatial complexity
  - Computational complexity
- Efficient SP model generation and solution needs special tools in
  - Modelling – generating model instance
  - Solving – using specialised algorithms
- New implementation doesn't suffer of the inherent problems of the previous

OptiRisk SYSTEMS

References

- Gay, David M. *Hooking your solver to AMPL*. Technical Report 93-10, AT&T Bell Laboratories, Murray Hill, NJ, 1993, revised, 1997.
- Gassmann, Horand I., and Eithan Schweitzer. "A comprehensive input format for stochastic linear programs." *Annals of Operations Research* 104.1-4 (2001): 89-125.
- Ellison, Francis, Gautam Mitra, and V. Zverovich. "FortSP: A stochastic programming solver." *OptiRisk Systems* (2010).