SARAH - A Tool for SUSY Model Builders

Florian Staub
University of Würzburg

arXiv:0806.0538
Comput.Phys.Commun.181:1077-1086,2010.
theorie.physik.uni-wuerzburg.de/~fnstaub/sarah.html

TOOLS 2010, Winchester
02. July 2010

Outline

Basic Idea

SARAH is a Mathematica package to get with minimal amount of information all properties of a ($\mathcal{N} = 1$)-SUSY-model

Basic Idea

SARAH is a Mathematica package to get with minimal amount of information all properties of a ($\mathcal{N} = 1$)-SUSY-model

Input: Gauge Groups, Particle Content, Superpotential

$\downarrow$

Lagrangian for Gauge Eigenstates

Basic Idea

SARAH is a Mathematica package to get with minimal amount of information all properties of a ($\mathcal{N} = 1$)-SUSY-model

Input: Gauge Groups, Particle Content, Superpotential

↓

Lagrangian for Gauge Eigenstates

↓

Input: Symmetry Breaking(s) and Rotations

↓

Final Lagrangian, Mass Matrices, Tadpole Equations

↓

Vertices, loop corrections, RGEs

Supported Models

SARAH can handle a large variety of models

Particle Content and Interactions

- Gauge sector can be any direct product of $SU(N)$ groups
- All irreducible representations of $SU(N)$ for chiral superfields are possible
- Matter interactions are defined in a compact form by superpotential
- All gauge interactions automatically added

- Arbitrary number of field rotations/symmetry breakings
- Gauge fixing terms can be specified in R_ξ gauge
- Non canonical terms can be added in component fields

Full control about parameters

All properties of each parameter can be defined separately:

- Allow/Forbid off-diagonal values
- Define if real or complex, symmetric/antisymmetric
- Define relations between parameters
- GUT normalization for gauge couplings
- Give TeX and output form

What happens automatically:

- Model checked for Gauge Anomalies
- Charge conservation of superpotential checked
- Soft SUSY Breaking terms are added
- Complete Lagrangian is calculated for component fields
- Ghost interactions are added
- Particles can be integrated out

  $\rightarrow$ Effective operators up to Dimension 6 calculated

Tree Level relations

- Masses / mass matrices are derived
- Tadpole equations for all VEVs are calculated

What can be calculated in addition:

- Specific vertices for given fields
- All vertices of the model: different lists for generic subclasses
- 1-loop corrections to Self Energies and Tadpoles

Renormalization Group Equations

1- and 2-loop anomalous dimensions of all superfields and all β-functions for

- gauge couplings
- superpotential parameters
- soft breaking parameters
- VEVs

Generations of fields can be treated as variable

Model Files

Model Files

Model files for CalcHep/CompHep and FenyArts/FormCalc can be generated.

Model Files

Model Files

Model files for CalcHep/CompHep and FenyArts/FormCalc can be generated.

Features of the model file for CalcHep/CompHep:

- Unitary and Feynman gauge supported
- Auxiliary fields for splitting of vertices with 4 colored particles included
- CP violation possible
- Can be used with MicrOmegas

LaTeX

A model specific LaTeX-file can be created:

- LaTeX-form of all parameters and fields can be defined in SARAH
- TeX output with all information about the model (particle content, mass matrices, tadpole equations, self-energies, RGEs, vertices)
- Including Feynman diagrams for all vertices using FeynMP
- New Mathematica functions for output of long formulas

Verification of the Output

Some of the checks so far:

- MSSM results checked against FeynArts and CalcHep/CompHep
- Relic density calculations with MicrOmegas were compared
- Analytical expressions of RGEs for MSSM, NMSSM and MSSM with broken R-Parity were compared with literature
- RGEs and loop corrections for the MSSM have been checked numerically against the implementation in `SPheno` ($\rightarrow$ later more)

Verification of the Output

Some of the checks so far:

- MSSM results checked against FeynArts and CalcHep/CompHep
- Relic density calculations with MicrOmegas were compared
- Analytical expressions of RGEs for MSSM, NMSSM and MSSM with broken R-Parity were compared with literature
- RGEs and loop corrections for the MSSM have been checked numerically against the implementation in `SPheno` ($\rightarrow$ later more)

Thanks to . . .

- Björn Hermann for checking the NMSSM
- Stefan Liebler for checking $\mu\nu$SSM
- Avelino Vincente for checking $SU(2)_L \times SU(2)_R$-Model

	CalcHep	SARAH
Ωh^2	0.191	0.191
Channels	38.72% $\chi_1\chi_1 \to e_3\bar{e}_3$	38.73% $\chi_1\chi_1 \to e_3\bar{e}_3$
	30.40% $\chi_1\chi_1 \to e_2\bar{e}_2$	30.39% $\chi_1\chi_1 \to e_2\bar{e}_2$
	29.23% $\chi_1\chi_1 \to e_1\bar{e}_1$	29.23% $\chi_1\chi_1 \to e_1\bar{e}_1$
	0.31% $\chi_1\chi_1 \to \nu_3\bar{\nu}_3$	0.31% $\chi_1\chi_1 \to \nu_3\bar{\nu}_3$
	0.30% $\chi_1\chi_1 \to \nu_2\bar{\nu}_2$	0.30% $\chi_1\chi_1 \to \nu_2\bar{\nu}_2$
	0.30% $\chi_1\chi_1 \to \nu_1\bar{\nu}_1$	0.30% $\chi_1\chi_1 \to \nu_1\bar{\nu}_1$
	0.24% $\chi_1\chi_1 \to u_2\bar{u}_2$	0.24% $\chi_1\chi_1 \to u_2\bar{u}_2$
	0.23% $\chi_1\chi_1 \to u_1\bar{u}_1$	0.23% $\chi_1\chi_1 \to u_1\bar{u}_1$
	0.10% $\chi_1\chi_1 \to d_3\bar{d}_3$	0.10% $\chi_1\chi_1 \to d_3\bar{d}_3$
	0.07% $\chi_1\chi_1 \to ZZ$	0.07% $\chi_1\chi_1 \to ZZ$
	0.04% $\chi_1\chi_1 \to d_2\bar{d}_2$	0.04% $\chi_1\chi_1 \to d_2\bar{d}_2$
	0.04%$\chi_1\chi_1 \to d_1\bar{d}_1$	0.04%$\chi_1\chi_1 \to d_1\bar{d}_1$

Included Model File

So far, the following models are included in the package

- MSSM: with/without flavor violation, in CKM basis, with CP violation
- NMSSM: with/without CP violation, in CKM basis
- $SU(5)$-extensions of the MSSM
- MSSM with bilinear R-parity violation and $\mu\nu$SSM
- SM, MSSM with additional $U(1)$, effective MSSM without gluino

Included Model File

So far, the following models are included in the package

- MSSM: with/without flavor violation, in CKM basis, with CP violation
- NMSSM: with/without CP violation, in CKM basis
- $SU(5)$-extensions of the MSSM
- MSSM with bilinear R-parity violation and $\mu\nu$SSM
- SM, MSSM with additional $U(1)$, effective MSSM without gluino

$\rightarrow$ Easy to create new models . . .

From MSSM to NMSSM

- The gauge sector hasn't to be changed

```
Gauge[[1]]={B, U[1], hypercharge, g1,False};
Gauge[[2]]={WB, SU[2], left, g2,True};
Gauge[[3]]={G, SU[3], color, g3,False};
```

From MSSM to NMSSM

- The gauge sector hasn't to be changed
- Add singlet superfield

```
Fields[[1]] = {{uL,dL}, 3, q, 1/6, 2, 3};
...
Fields[[5]] = {conj[dR], 3, d, 1/3, 1, -3};
Fields[[6]] = {conj[uR], 3, u, -2/3, 1, -3};
Fields[[7]] = {conj[eR], 3, e, 1, 1, 1};
```

From MSSM to NMSSM

- The gauge sector hasn't to be changed
- Add singlet superfield

```
Fields[[1]] = {{uL,dL}, 3, q, 1/6, 2, 3};
...
Fields[[5]] = {conj[dR], 3, d, 1/3, 1, -3};
Fields[[6]] = {conj[uR], 3, u, -2/3, 1, -3};
Fields[[7]] = {conj[eR], 3, e, 1, 1, 1};
Fields[[8]] = {Sing, 1, S, 0, 1, 1};
```

From MSSM to NMSSM

- The gauge sector hasn't to be changed
- Add singlet superfield
- Change superpotential

```
SuperPotential = {{{Yu,1},{q, Hu, u}},
        {{Yd,-1},{q, Hd, d}},{{Ye,-1},{l, Hd, e}},
        {{μ,1},{Hu, Hd}}};
```

From MSSM to NMSSM

- The gauge sector hasn't to be changed
- Add singlet superfield
- Change superpotential

```
SuperPotential = {{{Yu,1},{q, Hu, u}},
        {{Yd,-1},{q, Hd, d}},{{Ye,-1},{l, Hd, e}},
        {{λ,1},{Hu, Hd, S}},
        {{κ,1/3},{S,S,S}}};
```

From MSSM to NMSSM

- The gauge sector hasn't to be changed
- Add singlet superfield
- Change superpotential
- Give VEV to scalar singlet

```
DEFINITION[EWSB][VEVs]=
{{SHd0, {vd, 1/√2}, {sigmad, I/√2},{phid, 1/√2}},
{SHu0, {vu, 1/√2}, {sigmau, I/√2},{phiu, 1/√2}}};
```

From MSSM to NMSSM

- The gauge sector hasn't to be changed
- Add singlet superfield
- Change superpotential
- Give VEV to scalar singlet

```
DEFINITION[EWSB][VEVs]=
{{SHd0, {vd, 1/Sqrt[2]}, {sigmad, I/Sqrt[2]},{phid, 1/Sqrt[2]}},
{SHu0, {vu, 1/Sqrt[2]}, {sigmau, I/Sqrt[2]},{phiu, 1/Sqrt[2]}},
{SSing, {vS,1/Sqrt[2]},{sigmaS,I/Sqrt[2]},{phiS,1/Sqrt[2]}}};
```

From MSSM to NMSSM

- The gauge sector hasn't to be changed
- Add singlet superfield
- Change superpotential
- Give VEV to scalar singlet
- Change particle mixings

```
DEFINITION[EWSB][MatterSector]=
{{{SdL, SdR}, {Sd, ZD}},
...
{{phiu, phid}, {h, ZH}},
{{sigmau, sigmad}, {Ah, ZA}},
{{fB, fW0, FHd0, FHu0}, {L0, ZN}},
{{{fWm, FHdm}, {fWp, FHup}}, {{Lm,Um}, {Lp,Up}}}}
```

From MSSM to NMSSM

- The gauge sector hasn't to be changed
- Add singlet superfield
- Change superpotential
- Give VEV to scalar singlet
- Change particle mixings

```
DEFINITION[EWSB][MatterSector]=
{{{SdL, SdR}, {Sd, ZD}},
...
{{phiu, phid,phiS}, {h, ZH}},
{{sigmau, sigmad,sigmaS}, {Ah, ZA}},
{{fB, fW0, FHd0, FHu0,FSing}, {L0, ZN}},
{{{fWm, FHdm}, {fWp, FHup}}, {{Lm,Um}, {Lp,Up}}}}
```

From MSSM to NMSSM

- The gauge sector hasn't to be changed
- Add singlet superfield
- Change superpotential
- Give VEV to scalar singlet
- Change particle mixings
- Define properties of parameters (optional)

```
{κ,{LaTeX -> "\\kappa",
    Real -> True,
    Dependence -> None,
    Value -> None,
    LesHouches -> {EXTPAR,62} }}
```

From MSSM to NMSSM

- The gauge sector hasn't to be changed
- Add singlet superfield
- Change superpotential
- Give VEV to scalar singlet
- Change particle mixings
- Define properties of parameters (optional)

That's All!

The new model files are evaluated within few minutes

Evaluation time

Evaluation time in seconds (Intel Q8200, 2.33 GHz, 4GB Ram):

	MSSM	NMSSM	$\mu\nu$SSM	MSSM + $U(1)$
Lagrangian	12.75	19.02	27.06	16.14
Vertices	74.83	94.64	115.08	110.47
RGEs	50.72	91.07	265.29	68.18
Loop Corrections	7.07	8.14	7.98	8.84
FeynArts	0.74	1.12	0.98	0.48
CalcHep/CompHep	6.03	15.57	47.08	6.70
$\LaTeX$	0.81	1.25	1.38	1.48

The current situation

SARAH:

+ Easy to create new models
+ Fast way to get analytical expressions for RGEs, vertices, masses, . . .
- No routines for RGE running, phase space, . . .
- Numerically slow

The current situation

SARAH:

- + Easy to create new models
- + Fast way to get analytical expressions for RGEs, vertices, masses, ...
- - No routines for RGE running, phase space, ...
- - Numerically slow

SPheno:

- + Routines for RGE running, phase space integrals, loop integrals, ...
- + Numerically fast
- - Time consuming to implement new models

The current situation

SARAH:

+ Easy to create new models
+ Fast way to get analytical expressions for RGEs, vertices, masses, ...
- No routines for RGE running, phase space, ...
- Numerically slow

SPheno:

+ Routines for RGE running, phase space integrals, loop integrals, ...
+ Numerically fast
- Time consuming to implement new models

SPheno and SARAH

Is it possible to combine both programs?

The Idea

Define Model in SARAH

↓

Derive necessary information: couplings, masses, loop contributions, RGEs, decay channels, ...

↓

Generate model dependent source code for SPheno

↓

Compile and run SPheno

↓

Give masses/parameters to relic density calculator, event generator, ...

→ Automatized way from model building to phenomenology

Status

- Free parameters of the model and boundary conditions at different scales easy to define in SARAH

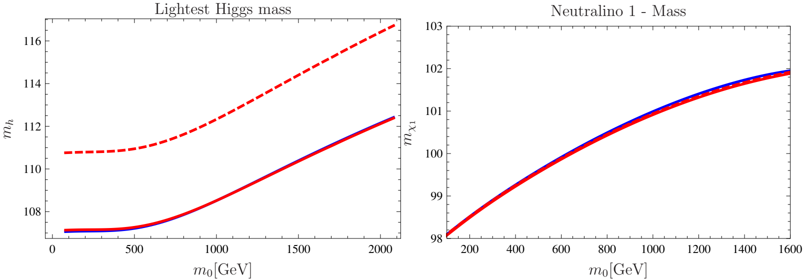
Completely general routines for source code generation of

- RGE running, even with thresholds: support of GUT theories
- All couplings
- Tree-level and 1-loop masses
- Two and three body decays
- In- and output

- Fit to electroweak data and low energy constraints adapted from SPheno

Check of the SPheno output

Comparison of a 'SARAH-SPheno' with SPheno 3v48



$$M_{1/2} = 250, A_0 = -300, \tan\beta = 10, \mu > 0$$

(dashed line: SPheno 3v48, red line: SPheno 3v48 without 2-loop Higgs masses, blue line: 'SARAH-SPheno')

SPheno output for other models (e.g. NMSSM, left/right model) tested at the moment

Summary

- SARAH is a tool for analyzing SUSY models
- Fast calculation of all tree level masses and vertices
- Routines for 1-loop self-energies/tadpoles and 1- and 2-loop RGEs
- Model files for CalcHep/CompHep and FeynArts/FormCalc can be created
- The models can be changed in SARAH in an easy and intuitive way
- Next version supposed to be a 'spectrum-generator-generator'

Mass matrices, vertices, . . .

- Loading SARAH and starting the model

```
<<path/SARAH.m
Start[''NMSSM'',''One_Rotation'']
```

Mass matrices, vertices, . . .

- Loading SARAH and starting the model
- All tree level masses are saved in Masses[Eigenstates]

```
Mass[VWm] /. Masses[EWSB]
```

Mass matrices, vertices, . . .

- Loading SARAH and starting the model
- All tree level masses are saved in `Masses[Eigenstates]`

$$(g_2^2 * (v_d^2 + v_u^2))/4$$

Mass matrices, vertices, . . .

- Loading SARAH and starting the model
- All tree level masses are saved in `Masses[Eigenstates]`
- Mass matrices a returned by `MassMatrix[particle]`

```
MassMatrix[hh][[3,3]]
```

Mass matrices, vertices, . . .

- Loading SARAH and starting the model
- All tree level masses are saved in Masses[Eigenstates]
- Mass matrices a returned by MassMatrix[particle]

$$m_s^2 + 3v_S^2\kappa^2 - v_d v_u \kappa\lambda + (v_d^2\lambda^2)/2 + (v_u^2\lambda^2)/2 + \sqrt{2} * v_S * T_\kappa$$

Mass matrices, vertices, ...

- Loading SARAH and starting the model
- All tree level masses are saved in `Masses[Eigenstates]`
- Mass matrices a returned by `MassMatrix[particle]`
- One specific vertex is calculated by `Vertex`

```
Vertex[{Ah,Ah,Ah,Ah}]
```

Mass matrices, vertices, . . .

- Loading SARAH and starting the model
- All tree level masses are saved in `Masses[Eigenstates]`
- Mass matrices a returned by `MassMatrix[particle]`
- One specific vertex is calculated by `Vertex`

$$\{\{\texttt{Ah[gt1]},\texttt{Ah[gt2]},\texttt{Ah[gt3]},\texttt{Ah[gt4]}\},\\ \{(I/4)(Z_A[\mathrm{gt1},2](4\lambda Z_A[\mathrm{gt2},3](Z_A[\mathrm{gt3},3](\kappa Z_A[\mathrm{gt4},1]- \\ \lambda Z_A[\mathrm{gt4},2]) + \ldots))),1\}\}$$

Mass matrices, vertices, ...

- Loading SARAH and starting the model
- All tree level masses are saved in `Masses[Eigenstates]`
- Mass matrices a returned by `MassMatrix[particle]`
- One specific vertex is calculated by `Vertex`
- Use `MakeVertexList[Eigenstates]` for calculating all vertices at once

Mass matrices, vertices, ...

- Loading SARAH and starting the model
- All tree level masses are saved in `Masses[Eigenstates]`
- Mass matrices a returned by `MassMatrix[particle]`
- One specific vertex is calculated by `Vertex`
- Use `MakeVertexList[Eigenstates]` for calculating all vertices at once
- Writing model files is done by `MakeFeynArts, MakeCHep[]`

Mass matrices, vertices, ...

- Loading SARAH and starting the model
- All tree level masses are saved in `Masses[Eigenstates]`
- Mass matrices a returned by `MassMatrix[particle]`
- One specific vertex is calculated by `Vertex`
- Use `MakeVertexList[Eigenstates]` for calculating all vertices at once
- Writing model files is done by `MakeFeynArts, MakeCHep[]`
- RGEs are calculated with `CalcRGEs[]`

Mass matrices, vertices, . . .

- Loading SARAH and starting the model
- All tree level masses are saved in `Masses[Eigenstates]`
- Mass matrices a returned by `MassMatrix[particle]`
- One specific vertex is calculated by `Vertex`
- Use `MakeVertexList[Eigenstates]` for calculating all vertices at once
- Writing model files is done by `MakeFeynArts, MakeCHep[]`
- RGEs are calculated with `CalcRGEs[]`

```
BetaYijk[[5,1]]
```

Mass matrices, vertices, . . .

- Loading SARAH and starting the model
- All tree level masses are saved in `Masses[Eigenstates]`
- Mass matrices a returned by `MassMatrix[particle]`
- One specific vertex is calculated by `Vertex`
- Use `MakeVertexList[Eigenstates]` for calculating all vertices at once
- Writing model files is done by `MakeFeynArts, MakeCHep[]`
- RGEs are calculated with `CalcRGEs[]`

λ

Mass matrices, vertices, . . .

- Loading SARAH and starting the model
- All tree level masses are saved in `Masses[Eigenstates]`
- Mass matrices a returned by `MassMatrix[particle]`
- One specific vertex is calculated by `Vertex`
- Use `MakeVertexList[Eigenstates]` for calculating all vertices at once
- Writing model files is done by `MakeFeynArts, MakeCHep[]`
- RGEs are calculated with `CalcRGEs[]`

BetaYijk[[5,2]]

Mass matrices, vertices, ...

- Loading SARAH and starting the model
- All tree level masses are saved in `Masses[Eigenstates]`
- Mass matrices a returned by `MassMatrix[particle]`
- One specific vertex is calculated by `Vertex`
- Use `MakeVertexList[Eigenstates]` for calculating all vertices at once
- Writing model files is done by `MakeFeynArts, MakeCHep[]`
- RGEs are calculated with `CalcRGEs[]`

$$-3g_1^2\lambda/5 - 3g_2^2\lambda + 2\kappa^2\lambda + 4\lambda^3 + \lambda(3\texttt{trace}[Y_d, \texttt{Adj}[Y_d]] + \texttt{trace}[Y_e, \texttt{Adj}[Y_e]] + 3\texttt{trace}[Y_u, \texttt{Adj}[Y_u]])$$

Mass matrices, vertices, ...

- Loading SARAH and starting the model
- All tree level masses are saved in `Masses[Eigenstates]`
- Mass matrices a returned by `MassMatrix[particle]`
- One specific vertex is calculated by `Vertex`
- Use `MakeVertexList[Eigenstates]` for calculating all vertices at once
- Writing model files is done by `MakeFeynArts, MakeCHep[]`
- RGEs are calculated with `CalcRGEs[]`
- `CalcLoopCorrections[Eigenstates]` calculates 1-loop self-energies and tadpoles

Mass matrices, vertices, ...

- Loading SARAH and starting the model
- All tree level masses are saved in `Masses[Eigenstates]`
- Mass matrices a returned by `MassMatrix[particle]`
- One specific vertex is calculated by `Vertex`
- Use `MakeVertexList[Eigenstates]` for calculating all vertices at once
- Writing model files is done by `MakeFeynArts, MakeCHep[]`
- RGEs are calculated with `CalcRGEs[]`
- `CalcLoopCorrections[Eigenstates]` calculates 1-loop self-energies and tadpoles

```
SelfEnergySum[VZ]
```

Mass matrices, vertices, ...

- Loading SARAH and starting the model
- All tree level masses are saved in Masses[Eigenstates]
- Mass matrices a returned by MassMatrix[particle]
- One specific vertex is calculated by Vertex
- Use MakeVertexList[Eigenstates] for calculating all vertices at once
- Writing model files is done by MakeFeynArts, MakeCHep[]
- RGEs are calculated with CalcRGEs[]
- CalcLoopCorrections[Eigenstates] calculates 1-loop self-energies and tadpoles

$$(|\Gamma_{Z,W^+,W^-}|^2(-8B_{22}(p^2,m^2_{W^-},m^2_{W^-}) - B_0(p^2,m^2_{W^-},m^2_{W^-}]) * (4p^2+2m^2_{W^-})))/2 + \ldots$$