Krzysztof Genser, Fermilab/SCD

17th Geant4 Collaboration Meeting
Chartres, September, 2012

Muon Stopping Discussion

Outline

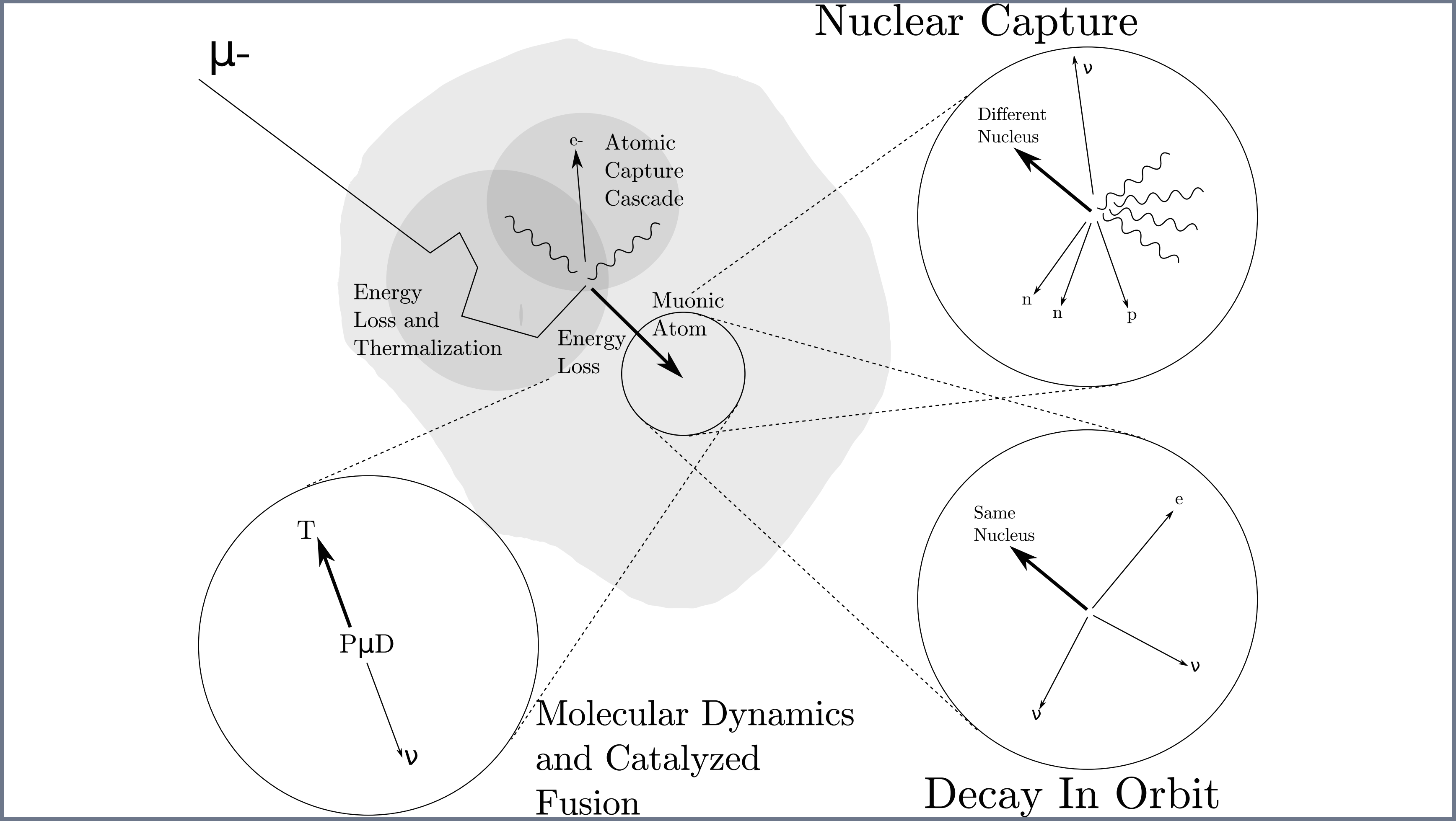
- Situation prior to 9.6beta release
- Recent History/Changes
  - Proposal by Kevin Lynch

    (within the context of Mu2e and other muon capture experiments)
  - Response/Changes by Vladimir Ivanchenko
  - Testing of the new code
  - Short and long term evolution suggestions
- Discussion

Recent Proposal by Kevin Lynch (now at CUNY/Mu2e)

- Mu2e is an experiment being designed at Fermilab
  - Mu2e is looking for neutrino-less muon-to-electron conversion in the Coulomb field of a nucleus (^{27}Al target)
  - simulating all known processes is very important for background calculations
- Kevin worked on PSI MuCap (mu^{-}+p) and MuSun (mu^{-}+d) experiments in the past and had created a prototype involving (possibly dynamically created) "transportable" Muonic Atoms (as G4ParticleDefinitions)
  - with factored out Atomic Capture, Nuclear Capture and Decay in Orbit (a.k.a. Bound Decay)
  - using the G4DecayChannel "technique" to "process" them
- The prototype is now part of the Mu2e offline framework (as an alternative to the standard Geant4 code)

Muon Capture Physics - artistic drawing from Kevin’s talk



Current “official” muon capture Geant4 implementations

- There are two main implementations:
  - G4MuonMinusCaptureAtRest, within hadronic framework, inheriting from G4VRestProcess, using
    - G4MuMinusCaptureCascade which does both the muon EM cascade and its (bound) decay in orbit
    - G4ExcitationHandler which does the nuclear de-excitation
  - G4QCapture
    - Alternative approach – outside hadronic framework

Response to Kevin's Proposal or Recent Changes by V. Ivanchenko*

- To address most of the current Mu2e needs (given that only heavy (~^{27}Al) isotopes are involved) Vladimir suggested (and had subsequently implemented) a revised code using a new, more modular and generic approach (and also addressed other needs within the hadronic framework)

- See next page for the description of G4MuonMinusCapture
  - the main muon capture class using the new approach

*) see e.g. Vladimir's talk at the Hadronic Group Meeting in April

G4MuonMinusCapture and related Changes

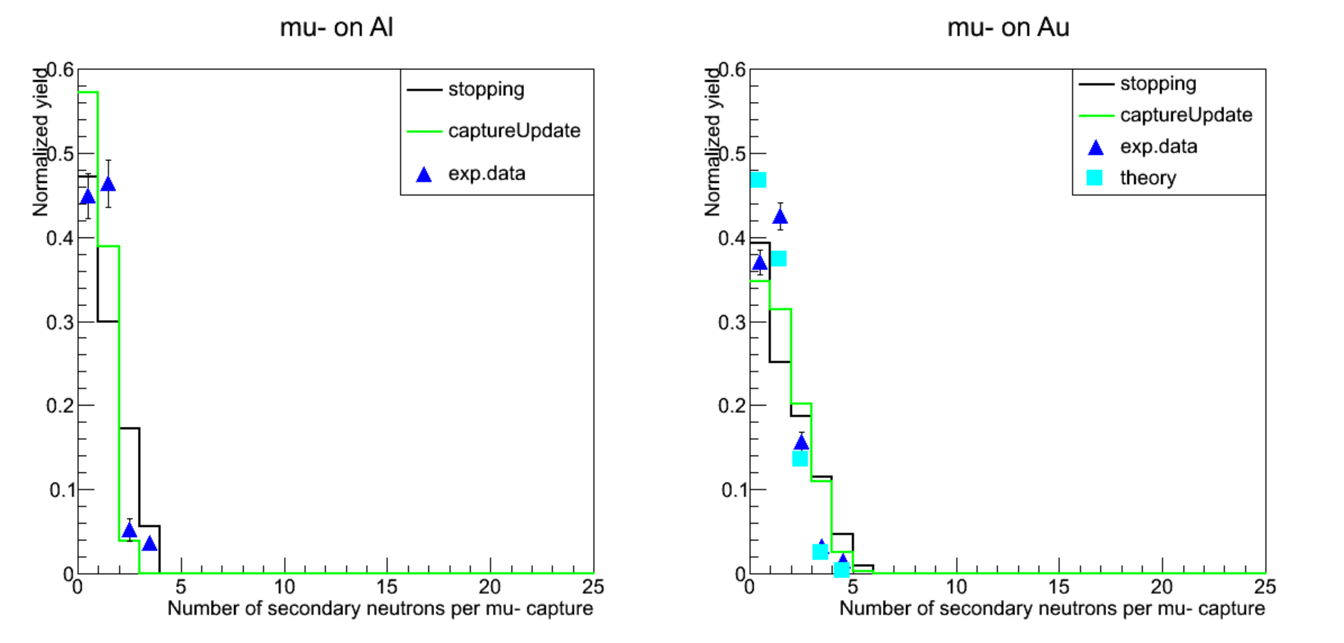
- G4MuonMinusCapture inherits from (newly added) G4HadronStoppingProcess which is a G4HadronicProcess which is a G4VDiscreteProcess (not a G4VAtRestProcess)

- G4HadronStoppingProcess
  - defines (among others) AtRestDoIt (where most of the work is done) as well as SetEmCascade and SetBoundDecay functions
  - it uses G4EmCaptureCascade, G4MuonMinusBoundDecay (when made available) and G4MuMinusCapturePrecompound which are classes inheriting from G4HadronicInteraction
  - also available now to be used in other applicable places

- In the constructor G4MuonMinusCapture now takes a pointer to G4VPrecompundModel (which defaults to G4PrecompoundModel which is another G4HadronicInteraction) passed to G4MuMinusCapturePrecompound constructor (which G4MuonMinusCapture also registers with G4HadronicProcessStore)

Testing of G4MuonMinusCapture

- commissioning/test (test48) results (9.6.b01 + updates) (by/with Julia Yarba)
  - updated code is working and produces similar results to the old stopping code with a decent agreement with the data/theory



data from review paper: P. Singer, Springer Tracts in Modern Physics, 71, 39 (1974)

  - coding remark: we had asked authors of CLHEP to make LorenzVector single argument constructor to be declared explicit (a generally good idea)

Evolving G4MuonMinusCapture

- Building upon the foundation created by Vladimir, we would further suggest to modify G4MuonMinusCapture (or the relevant parts of the infrastructure) to allow for
  - inclusion of experimental data for specific isotopes, e.g. ^{27}Al
  - selectively turning off the nuclear capture/bound decay (EM cascade?)
    - e.g. by setting their pointers to null?

Evolving G4MuonMinusCapture

- e.g. we would suggest to
  - modify G4MuonMinusCapture constructor to take a different argument
  - to allow to replace G4MuMinusCapturePrecompound with an alternative version

Evolving G4MuonMinusCapture

i.e. replace:

```
G4MuonMinusCapture(G4VPreCompoundModel* ptr=0);

G4MuonMinusCapture::G4MuonMinusCapture(G4VPreCompoundModel* ptr)
  : G4HadronStoppingProcess ("muMinusCaptureAtRest")
{
  SetBoundDecay(new G4MuonMinusBoundDecay());
  RegisterMe(new G4MuMinusCapturePrecompound(ptr));
}
```

with:

```
explicit G4MuonMinusCapture(G4HadronicInteraction* hiptr=0);

G4MuonMinusCapture::G4MuonMinusCapture(G4HadronicInteraction* hiptr)
  : G4HadronStoppingProcess ("muMinusCaptureAtRest")
{
  SetBoundDecay(new G4MuonMinusBoundDecay());
  RegisterMe(hiptr); // G4HadronicProcess function registering generator of secondaries with the H.I. Registry
}
```

and then use it e.g. like:

```
theProcess = new G4MuonMinusCapture(new UserMuMinusCapturePrecompound());  // see next page
```

Evolving G4MuMinusCapturePrecompound

- where UserMuMinusCapturePrecompound() is a version of G4MuMinusCapturePrecompound modified e.g. to accommodate special cases for specific isotopes
- It (also, as did its predecessor) takes and optional pointer to G4VPreCompoundModel as an argument
- Could it inherit from G4MuMinusCapturePrecompound ?
  - after appropriate modifications/restructuring, also to to allow reusing algorithms in more than one special case?

Longer Term Changes

- Following ideas from Kevin’s original proposal in order to address the needs of other muon capture experiments (with lighter targets) we would suggest to further extend the toolkit:
  - Introduce “transportable” light muonic atoms and molecules
    - H, D, T, He3, and He4, P-mu-P,…
      - include proper spin treatment
    - They all could be pre-created in the PreInit phase not necessitating a dynamic creation and related complications
    - Could they inherit, in some way, from the corresponding G4Ions?
- Could Kevin’s prototype be the basis of the extension?
  - What would be the best way to handle the new exotic states?
    - What is the best moment to produce them?

Discussion

- Is the outlined modification/evolution approach reasonable/optimal?
- Are there other comments/suggestions?