



Steffen Schwigon  
<[schwigon@webit.de](mailto:schwigon@webit.de)>  
Dresden Perl Mongers

# Die Gebrüder Yaswi



*Language::Prolog::Yaswi*

*Language::Prolog::Types*

*Language::Prolog::Sugar*

# WiesoWeshalbWarum

- SWI-Prolog ist ein freies Prolog-System
- Manchmal Problem in Prolog eleganter lösbar
- Language::Prolog::Yaswi ist  
*Yet Another interface to SWI-Prolog*
- Mappt Abfragen und Datenstrukturen zwischen Perl und Prolog
  - beide Richtungen möglich: Perl -> Prolog -> Perl -> ...

# Die Module

- Language::Prolog::Yaswi
  - der eigentliche Wrapper zu SWI-Prolog

# Die Module

- `Language::Prolog::Yaswi`
  - der eigentliche Wrapper zu SWI-Prolog
- `Language::Prolog::Types`
  - damit können z.B. Fehler-Ausgaben, die vom Prolog kommen, natürlich lesbar gemacht werden, anstelle lustiger `HASH(0x814e684)`-Ausgaben

# Die Module

- `Language::Prolog::Yaswi`
  - der eigentliche Wrapper zu SWI-Prolog
- `Language::Prolog::Types`
  - damit können z.B. Fehler-Ausgaben, die vom Prolog kommen, natürlich lesbar gemacht werden, anstelle lustiger `HASH(0x814e684)`-Ausgaben
- `Language::Prolog::Sugar`
  - erlaubt das transparente Verwenden von Prolog-Termen und Variablen mitten im Perl-Quellcode

# Die Module

- `Language::Prolog::Yaswi`
  - der eigentliche Wrapper zu SWI-Prolog
- `Language::Prolog::Types`
  - damit können z.B. Fehler-Ausgaben, die vom Prolog kommen, natürlich lesbar gemacht werden, anstelle lustiger `HASH(0x814e684)`-Ausgaben
- `Language::Prolog::Sugar`
  - erlaubt das transparente Verwenden von Prolog-Termen und Variablen mitten im Perl-Quellcode
- **einfach immer alle verwenden; von SYNOPSIS klauen**

# Prolog in Perl einbinden

- Prädikat „terminvu“
- liefert mögliche Termine zu irgendwas

```
terminvu(date(Y,M,D)) :-  
    epoch_calendar(date(Y,M,D), T),  
    terminvu_epoch(T).
```

# Prolog in Perl einbinden

- Interaktiv in Prolog-Shell:

```
?- terminvu(Termin) .  
Termin = date(1970, 1, 12) ;  
Termin = date(1970, 1, 13) ;  
Termin = date(1970, 1, 14) ;  
Termin = date(1970, 1, 15) ;  
Termin = date(1970, 1, 16) ;  
Termin = date(1970, 1, 19) ;  
Termin = date(1970, 1, 20) ;  
No  
?-
```

# Prolog in Perl einbinden

- Jetzt Perl:
  - `Language::Prolog::*` Module laden
  - Prädikate und Variablen deklarieren, die man transparent verwenden möchte
  - Prolog-Code laden ('consult'en)
  - Query setzen und Ergebnisse abfragen

# Prolog in Perl einbinden

```
use Language::Prolog::Types ':short';
use Language::Prolog::Types::overload;
use Language::Prolog::Yaswi qw(:query :assert :load);

use Language::Prolog::Sugar
  functors => {
    terminvu => 'terminvu',
  },
  vars     => [qw( Termin )],
;

swi_consult ('calendar.swipl');
swi_set_query(terminvu(Termin));
while (swi_next) {
  print sprintf(" Terminvu = %s\n", swi_var(Termin));
}
```

# Andere Richtung

- Grade hatten wir Prolog im Perl
- Jetzt verwenden wir Perl im Prolog ...

# Perl in Prolog einbinden

- Perl-Code:

```
package EfOnline::Terminplaner::SwiSupport ;

sub date2epochday {
    my ($year, $month, $day) = @__;
    Date::Calc::Date_to_Days($year,
                             $month,
                             $day);
}
```

# Perl in Prolog einbinden

- Prolog-Code:

```
:- module(calendar, [ ]).  
:- perl5_eval('use EfOnline::Terminplaner::SwiSupport', _).  
date2epochday(date(Year,Month,Day), Epochday) :-  
    perl5_call('EfOnline::Terminplaner::SwiSupport::date2epochday',  
              [Year, Month, Day],  
              [Epochday]).
```

# Letzte**K**onsequenz

- Schicker Perl-Prolog-Cyborg ...

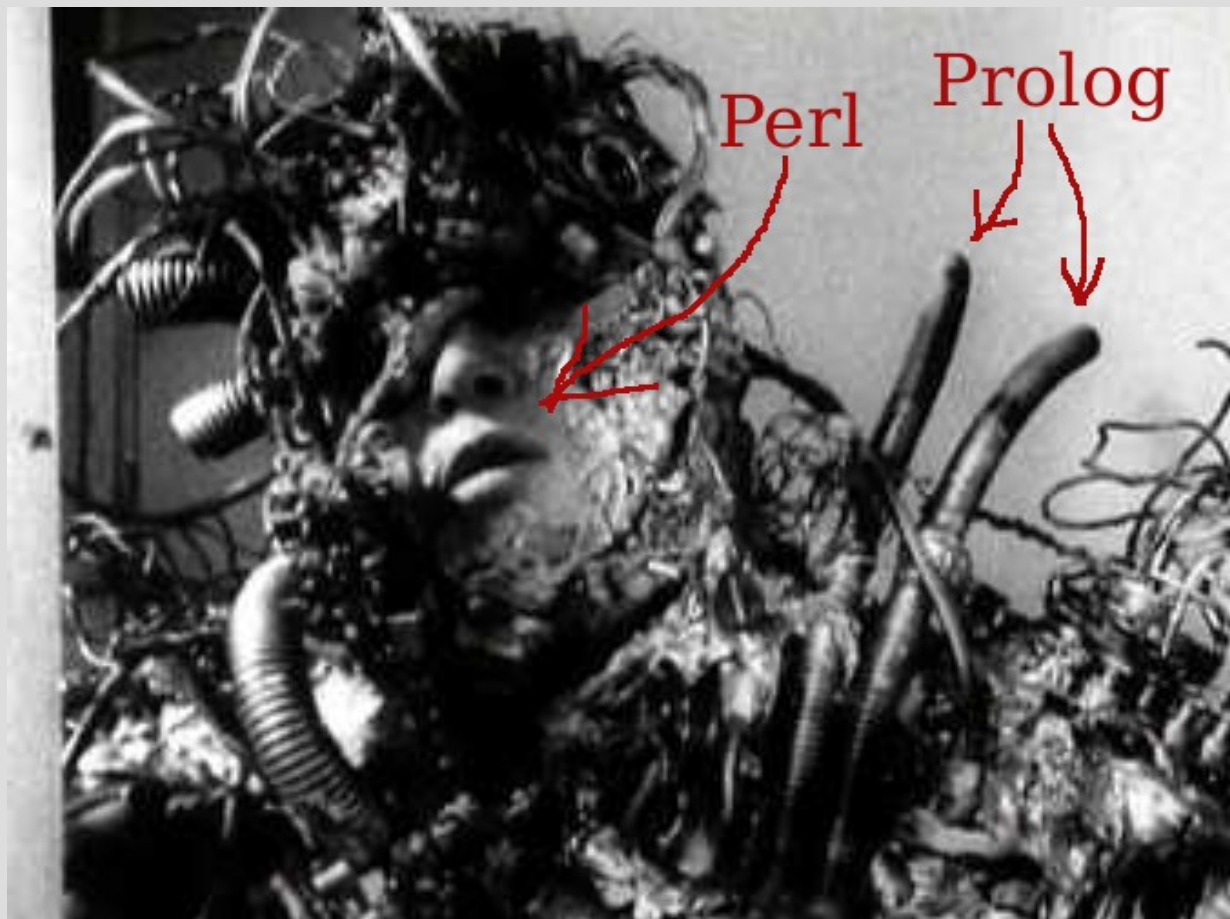
# Letzte Konsequenz

- Schicker Perl-Prolog-Cyborg ...



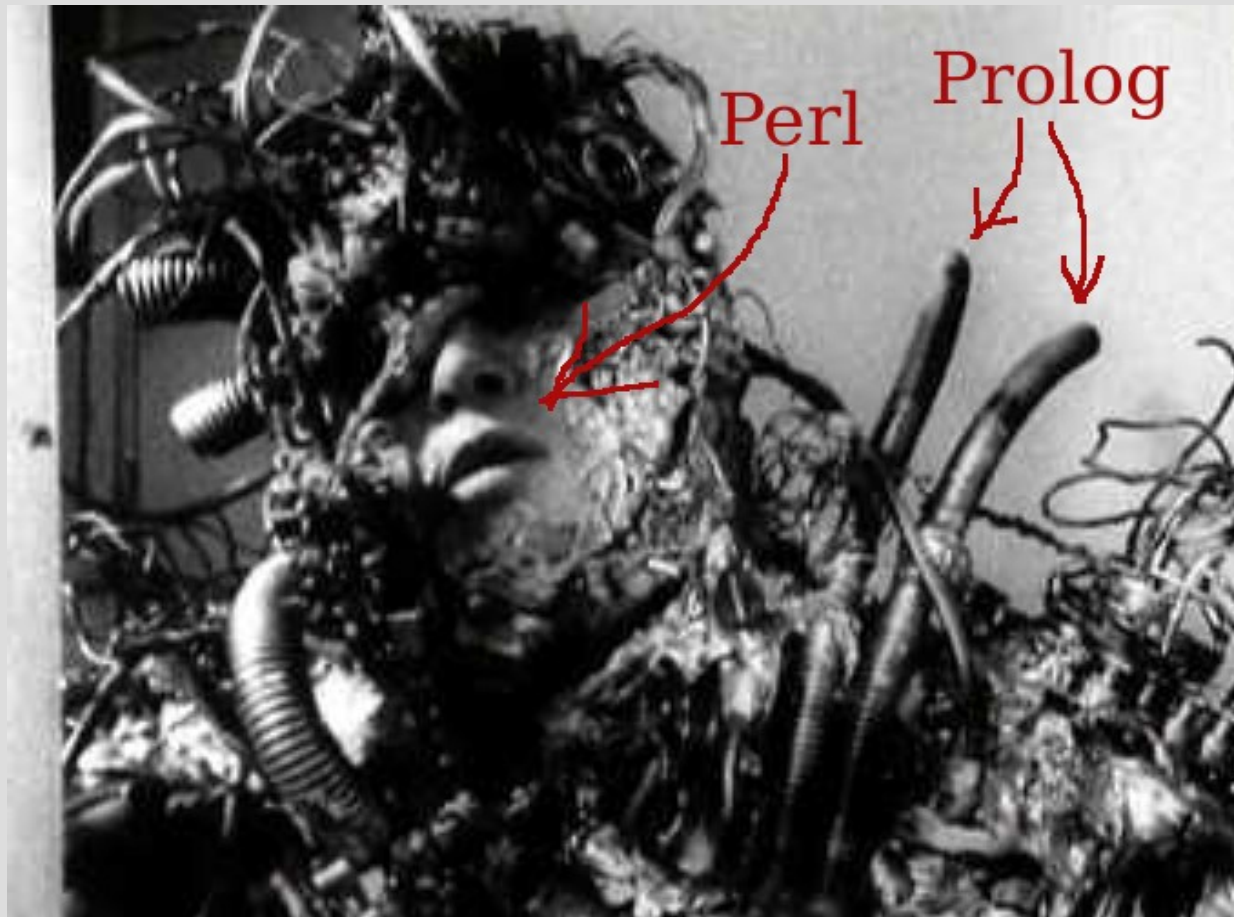
# Letzte Konsequenz

- Schicker Perl-Prolog-Cyborg ...



# Letzte Konsequenz

- Schicker Perl-Prolog-Cyborg ...



ENDE.