There is a rather confusing feature in Specman's coverage engine that I would like to share with you today. I've met several people (including myself) who had been struggling to understand what was going on there and gave up Recently I was called to the rescue again with the same problem so I guess it's a good opportunity to tell you guys about it.

So imagine you have a struct called packet. And in the packet you have a **length** field which could be anything from 0 to 255. You're doing a lot of things with this packet in your environment and you also want to add some coverage. Let's see some code:

{code}

struct packet {

length: byte;
 event cover_me;
 cover cover_me is {
 item length;
 };

};

{/code}

Very simple so far. Next we want to limit the coverage spectrum of this item because we're not interested in values over 100. We will use the **ignore** command for that:

{code}

item length using ignore = length > 100;

{/code}

For the less experienced guys out there - note that the coverage commands **ignore** and **when** may look as two alternatives for limiting the coverage collection, but in fact they are fundamentaly different from each other and should be used for different purposes. The **ignore** command is used to narrow down the coverage spectrum while the **when** command is used to narrow down the number of coverage collection occurrences.

Back to our business, after we've narrowed down the coverage spectrum to values below 100 only, we want to have an additional limitation, and ignore values under 90. Let's do this:

{code}

item length using ignore = length > 100, ignore = length < 90;

{/code}

You think this is going to work? Not really. The code will compile and run successfully but the coverage engine will only take into consideration the last ignore command. Really disappointing. This problem typically arises with more complex items such as cross items.

Now for the good news: the workaround is to write all your ignores in one (long) line Not so comfortable with complex items, but it's the only way it will work. Also, make sure to use **or** (and not and) as a separator between adjacent ignore conditions because we're dealing with inverse logic here.

So let's conclude with the full example again, and a few lines of code that demonstrate it (you're gonna have to open the coverage window to see this).

Here we go:

```
{code}
struct packet {
 length: byte;
 event cover_me;
 cover cover_me is {
 item length using
 // the 2 lines below will NOT do the job
 //ignore = length > 100, // this ignore will be ignored !!
 //ignore = length < 90;
 // the line below WILL do the job
 ignore = length > 100 or length < 90;
 };
 };


extend sys {
 !packet: packet;
```

```
 run() is also {
 for i from 1 to 100 {
 gen packet;
 emit packet.cover_me;
 };
 };
 };
```

{/code}