Práctica 206. Configuración

Modelos simples de razonamiento de configuración.

Definiciones básicas

La configuración es, hoy por hoy, el área donde los SBC han conseguido mayores éxitos desde el punto de vista de su implantación industrial. Por ejemplo, podemos citar los sistemas R1/XCON y XSEL, para la configuración de ordenadores de la empresa Digital (hoy HP), o el sistema PROSE, para la configuración de sistemas de telefonía digital.

La configuración es la actividad que a partir de un conjunto predeterminado de *componentes*, cada uno de los cuales vendrá descrito por un conjunto predeterminado de *parámetros* con sus correspondientes valores y de un conjunto predeterminado de posibles *puertos* para relacionar los componentes, produce un *artefacto* que satisface ciertos *requisitos*.

Es decir, el proceso de configurar un artefacto parte de la siguiente información:

- ¿Qué condiciones se le exigen al artefacto? (requisitos)
- ¿De qué elementos se puede componer el artefacto? (componentes)
- ¿Cuáles son los atributos relevantes de cada componente, así como sus posibles valores? (parámetros)
- ¿Cuáles son las relaciones relevantes entre los componentes, así como sus posibles valores? (puertos–conectores)

A partir de esta información, se van seleccionando los elementos componentes y sus parámetros y se van organizando, a fin de satisfacer los requisitos. Algunos ejemplos de problemas de configuración pueden ser los anteriormente citados relativos a equipos informáticos o telemáticos. Otros más cotidianos podrían ser hallar una dieta para una persona a partir de sus necesidades y preferencias y de un conjunto de platos; colocar un conjunto de muebles en una habitación de forma que se satisfagan todas las necesidades de funcionalidad;... Es obvio que los problemas de este tipo son más complejos que los de clasificación.

Razonamiento mediante satisfacción de restricciones

Como es sabido, un problema de satisfacción de restricciones queda definido por los siguientes elementos:

- Un conjunto finito de variables $\{X_1, \ldots, X_n\}$.
- Para cada variable X_i, un conjunto finito D_i llamado dominio.
- Un conjunto finito de restricciones $\{C_1, \ldots, C_p\}$. Cada restricción C_j viene dada por un subconjunto no vacío de variables $\{X_{j,1}, \ldots, X_{j,a_j}\}$ y un conjunto $V_j \subseteq D_{j,1} \times \ldots \times D_{j,a_j}$.

El valor a_j es la aridad de la restricción y el conjunto V_j es el conjunto de asignaciones parciales que satisfacen la restricción. Habitualmente, una restricción

C_j no viene dada explícitamente por el conjunto V_j, sino por una expresión booleana que para cada asignación devolverá "verdadero" o "falso".

Se llama asignación a una función que a cada variable le asigna un valor de su dominio. Resolver un problema es encontrar una asignación total que satisfaga todas las restricciones.

Muchos problemas de configuración se pueden expresar de forma directa como satisfacción de restricciones. Por ejemplo:

Se dispone de 5 dispositivos, A, B, C, D, E, descritos en la tabla adjunta. Queremos configurar un artefacto formado por un subconjunto de estos dispositivos tal que el coste total sea menor de 23 y cumpla las tres funcionalidades f_1, f_2, f_3.

	precio	función	incompatible-con	exige
A	10	f_1	C	B
B	5	f_2	-	A
C	4	f_3	A	E
D	20	f_1, f_2	E	-
E	7	f_2, f_3	D	-

Es fácil identificar en este problema:

–Cinco variables A, B, C, D, E con el dominio común {ausente, presente}.

–Las restricciones binarias dadas por las incompatibilidades y exigencias de las dos últimas columnas.

–Las restricciones de aridad variable que expresan que la suma de precios de los componentes presentes es menor que 23 y que la unión de funciones de los componentes presentes es $\{f_1, f_2, f_3\}$.

Muchas de estas restricciones (por ejemplo, la incompatibilidad de componentes, o el coste total inferior a 23) pueden calcularse en las asignaciones parciales, podando así muchas ramas del árbol de búsqueda. Si las variables se asignan en el orden indicado, el lector puede comprobar que se generan únicamente 8 de las 32 asignaciones totales, y de estas 8 solamente una satisface la restricción de funcionalidad (A, B, E presentes, C, D ausentes).

El problema de este enfoque radica en la *explosión combinatoria*, que surge porque el número de configuraciones posibles crece exponencialmente con el número de componentes y parámetros. Una búsqueda con retroceso puede ser por tanto inabordable, aún cuando se empleen ciertos heurísticos para ordenar las variables y sus valores, y aún cuando se empleen algoritmos más "astutos", como los de consistencia en arcos, para simplificar la búsqueda. Por ello, en los casos complejos es necesario emplear conocimiento heurístico más sofisticado y dependiente del problema para guiar el proceso de generación de las configuraciones.

Razonamiento mediante reglas heurísticas de producción

La primera propuesta para superar la explosión combinatoria fue el uso de reglas de producción (como las conocidas de CLIPS) de forma que cada regla correspondía a un paso o decisión tomada en el proceso de configuración. Esta es la idea fundamental del ya mencionado sistema XCON ([3]), para configuración de ordenadores Vax. Un buen resumen del razonamiento seguido por XCON puede encontrarse en [5], pp. 625-633. Aquí damos un resumen grosero e inexacto, pero que puede servir de guía para el desarrollo de SBC basados en estos principios.

A partir del conocimiento extraído a los humanos que realizaban esta tarea, y empleando algunos "trucos", fue posible implementar un sistema con encadenamiento hacia adelante que a partir de la especificación inicial y el catálogo de componentes va añadiendo unas y otros hasta llegar a una solución completa, sin realizar tanteos ni retrocesos. Para ello:

1. Se organiza el razonamiento en varias fases consecutivas, a saber:
   a) Completar la especificación, sustituyendo o añadiendo componentes requeridos.
   b) Disponer adecuadamente los componentes
   c) Generar el *layout* y el cableado.
2. Cada fase está compuesta de diversas *subtareas*, implementada cada una de ellas mediante una o varias reglas de producción.
3. La selección y ordenación temporal de estas subtareas es diferente en cada problema; por ello, las reglas de producción deben contener en su parte izquierda las condiciones que aseguren su aplicabilidad en un problema dado. Ello se consigue preguntando si ciertos valores han sido ya asignados, y codificando de alguna forma lo que equivale a una "búsqueda con previsión" en términos de algoritmos de satisfacción de restricciones.

Aclaremos esto con un ejemplo de un dominio menos técnico, adaptado de [6]: supongamos que queremos realizar la compra en el supermercado y que recibimos la siguiente orden: "Traer comida para el desayuno y unas cervezas". La primera fase (a) consistirá en generar a partir de esto una lista completa de la compra, a partir de las restricciones del mandante y de la tienda, lo cual puede llevar a reglas como

```
;;refinar el componente ‘‘desayuno’’
SI desayuno es requerido
ENTONCES añadir cartón-leche es requerido Y
    añadir bote-cafe-soluble es requerido Y
    añadir paquete-galletas es requerido Y
    quitar desayuno es requerido.

;;añadir un componente no especificado inicialmente
SI pack-latas-cerveza es requerido Y papas-fritas es en-oferta
ENTONCES añadir bolsa-papas-fritas es requerido.
```

La segunda fase (b) consistirá en disponer en bolsas los objetos requeridos y comprados, lo cual puede expresarse mediante reglas como

```
;;colocar un objeto pesado en una nueva bolsa.
SI X es requerido Y X no está colocado Y X es pesado
ENTONCES añadir B = bolsa nueva Y
    añadir X está colocado en B
```

El problema de este enfoque, el típico de los primeros sistemas expertos, es la *fragilidad* de los sistemas resultantes, consistente en lo siguiente:

1. Una base de reglas, según se suele decir, representa el *conocimiento* del dominio. Pero esto no es enteramente verdad, pues una base de reglas no

puede emplearse para realizar dos tareas que empleen el mismo conocimiento pero con finalidad levemente diferente; por ejemplo, si las reglas sirven para generar una solución, no podrán emplearse para comprobar si una propuesta es realmente una solución.

2. Una base de reglas resulta en general muy grande y muy difícil de depurar y mantener, ya que las reglas mezclan y representan de manera engañosamente uniforme el conocimiento cierto acerca de los componentes y sus parámetros, el conocimiento acerca de las especificaciones, y el conocimiento heurístico que simplifica la búsqueda. Como ejemplo, citemos que en 1989 XCON constaba de unas 15.000 reglas ([1]).

Modelos avanzados de razonamiento de configuración

Espacio de configuración y espacio de especificaciones

Como ya hemos mencionado, al plantear el problema de configuración como una satisfacción de restricciones, se pierde casi por completo la estructuración que un ser humano tendría en cuenta a la hora de resolver el problema; y al representar uniformemente mediante reglas de producción todo el conocimiento de configuración, se pierde de vista en qué consiste exactamente este conocimiento. Por tanto, será más conveniente considerar explícitamente y distinguir en un primer nivel de análisis y diseño varias tareas diferentes de razonamiento, cada una de las cuales se implementará con los algoritmos o procedimientos que en cada caso resulten más oportunos. Desde un punto de vista general, y siguiendo la estrategia llamada "proponer-y-revisar" (*propose-and-revise*), estas tareas son las siguientes:

- "Hacer operativos" los requisitos iniciales, comprobando su consistencia y traduciéndolos a un lenguaje formal de especificación. Para ello será necesario a veces emplear conocimiento declarativo acerca del dominio y llevar a cabo sesiones de extracción de conocimiento. El resultado de ello será un conjunto inicial de especificaciones (restricciones, que habrán de satisfacerse en todo caso, y preferencias, que deberán respetarse en lo posible.)

- "Inicializar" la configuración. A partir de las especificaciones y del conocimiento declarativo, se genera un primer conjunto de componentes con sus parámetros y sus relaciones (estructura.)

- "Proponer" una modificación a la configuración existente. Esta modificación será un refinamiento de componentes o el añadido de nuevas instancias de componentes.

- "Verificar y criticar" la modificación, es decir, comprobar si viola alguna restricción y en caso afirmativo proponer una acción que evite tal violación. Para que esto sea eficaz suele ser necesario recurrir al llamado "conocimiento de arreglos" (fix knowledge) que indica qué acciones son más apropiadas para cada situación de violación.

- "Modificar" la configuración en función de la acción seleccionada, y en su caso también las especificaciones o restricciones. Una causa para esto último es precisamente la aparición de nuevos componentes o el refinamiento de los existentes. Otra, más profunda, es el descubrimiento de que

las especificaciones iniciales eran ambiguas, o imposibles o muy difíciles de satisfacer; si es necesario modificar las especificaciones por esta última causa, estaremos en realidad ante un problema de *diseño*, no de configuración.

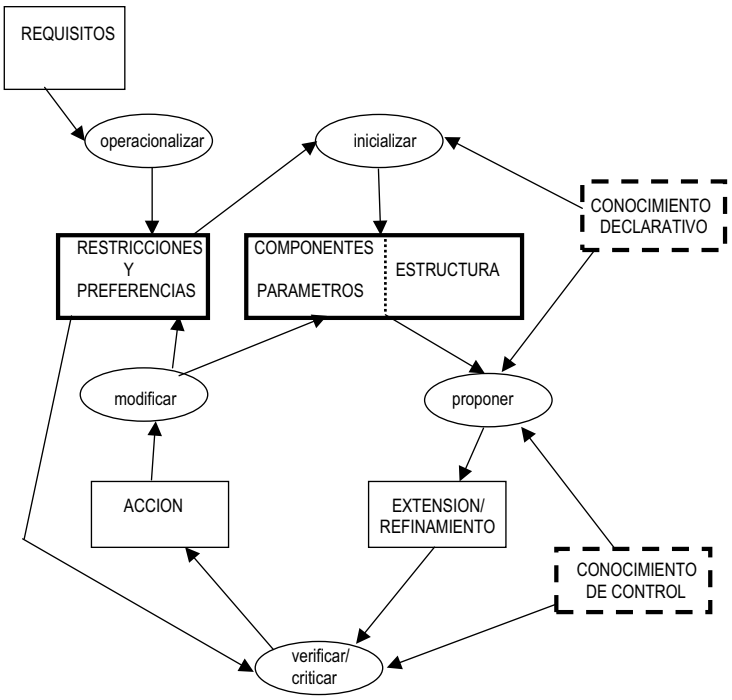

Figura 1: Razonamiento de configuración.

Por tanto, en general puede considerarse que la configuración se desarrolla en dos espacios, el de las *especificaciones* y el de la propia configuración, cada uno de ellos con cierta estructura. Todo esto se esquematiza en la figura 1, de elaboración propia a partir de varias que aparecen en [5], [4] y [2].

Razonamiento basado en casos

Este enfoque, radicalmente diferente de los anteriores, supone que se dispone previamente de una biblioteca B de *casos* previamente resueltos, consistente en un conjunto de pares *C(requisitos, solución)*. Dado un nuevo problema, es decir, unos nuevos requisitos R, el procedimiento puede describirse como sigue:

1. Hallar el caso $C_i(r_i, s_i)$ tal que la distancia de r_i a R sea mínima. Nótese que ello presupone que se ha definido una métrica adecuada en el conjunto de los requisitos.

2. Modificar s_i de forma que satisfaga todos los requisitos R. Nótese que ello supone definir los adecuados operadores de modificación de la solución y una estrategia (a ciegas o, mejor, heurística) que los vaya eligiendo y aplicando hasta conseguir el objetivo.

Si la biblioteca de casos resueltos es grande y cubre uniformemente el espacio de posibles problemas, y hay métricas claras y heurísticos eficientes, este método es muy efectivo y puede evitar casi por completo la búsqueda. En otro caso, la fase (2) del procedimiento será, en realidad, una instancia del problema de configuración tan complicada o más que la original.

Referencias

[1] V. Barker y D. O'Connor. Expert systems for configuration at digital: XCON and beyond. *Communications ACM*, 32(3):298–318, 1989.

[2] Joost Breuker y Walter van de Velde, (eds.). *CommonKADS Library for Expertise Modelling.* IOS Press, Amsterdam, 1994.

[3] John P. McDermott. R1: A rule-based configurer of computer systems. *Artificial Intelligence*, 19(1):39–88, 1982.

[4] Guus Schreiber, Hans Akkermans, Anjo Anjewieden, Robert de Hoog, Nigel Shadbolt, Walter van de Velde, y Bob Wielinga. *Knowledge Engineering and Management. The CommonKADS Methodology.* MIT Press, Cambridge, Ma., 2000.

[5] Mark Stefik. *Introduction to knowledge systems.* Morgan Kaufmann, Los Altos, Ca., 1995.

[6] Patrick Henry Winston. *Artificial Intelligence.* Addison-Wesley, Reading, Ma., 3rd ed., 1992.